Understanding how Knowledge is exploited in Ant Algorithms

Thomas Edward Reid McCallum

Doctor of Philosophy
Artificial Intelligence Applications Institute
School of Informatics
University of Edinburgh

2005

Abstract

Ant algorithms were first written about in 1991 and since then they have been applied to many problems with great success. During these years the algorithms themselves have been modified for improved performance and also been influenced by research in other fields. Since the earliest Ant algorithms, heuristics and local search have been the primary knowledge sources. This thesis asks the question "how is knowledge used in Ant algorithms?"

To answer this question three Ant algorithms are implemented. The first is the Graph-based Ant System (GBAS), a theoretical model not yet implemented, and the others are two influential algorithms, the Ant System and Max-Min Ant System. A comparison is undertaken to show that the theoretical model empirically models what happens in the other two algorithms. Therefore, this chapter explores whether different pheromone matrices (representing the internal knowledge) have a significant effect on the behaviour of the algorithm. It is shown that only under extreme parameter settings does the behaviour of Ant System and Max-Min Ant System differ from that of GBAS.

The thesis continues by investigating how inaccurate knowledge is used when it is the heuristic that is at fault. This study reveals that Ant algorithms are not good at dealing with this information, and if they do use a heuristic they must rely on it relating valid guidance. An additional benefit of this study is that it shows heuristics may offer more control over the exploration-exploitation trade-off than is afforded by other parameters.

The second point where knowledge enters the algorithm is through the local search. The thesis looks at what happens to the performance of the Ant algorithms when a local search is used and how this affects the parameters of the algorithm. It is shown that the addition of a local search method does change the behaviour of the algorithm and that the strength of the method has a strong influence on how the parameters are chosen.

The final study focuses on whether Ant algorithms are effective for driving a local search method. The thesis demonstrates that these algorithms are not as effective as some simpler fixed and variable neighbourhood search methods.

Acknowledgements

Many thanks to John Levine for his support and advice. Thanks to Austin Tate and Jim Blythe for their advice. Thanks to Dave Robertson for taking the time to read and to advise me on this work.

Many thanks to Catriona Tullo and Ronan Daly for proof reading chapters. Finally, thanks to my family for their endless support.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Thomas Edward Reid McCallum*)

To me it seems that those sciences are vain and full of error which are not born of experience, mother of all certainty, first hand experience which in its origins, or means, or end has passed through one of the five senses.

Leonardo da Vinci - "The Archetype of Human Potentia"

Table of Contents

Part I

Background

Chapter 1

Introduction

The primary question answered by this thesis is "what is the role of knowledge in Ant algorithms?" To answer this question several smaller questions must be asked:

- What is knowledge in this context?
- What are Ant algorithms?
- What knowledge sources does an Ant algorithm make use of?
- How is knowledge fed into the algorithm?
- What happens if this knowledge is misleading?
- How does this knowledge affect the behaviour of the Ant algorithms?

Knowledge in this context refers to two types of information. The first is derived from the process of the algorithm, this is termed *internal knowledge*. The second type of knowledge is called *external knowledge*. This type of knowledge is that which is given by either the user or by some other process outside of the algorithm.

Ant algorithms are biologically inspired algorithms that have become very popular since 1991 when the first papers on them were published. Since then the area has grown and many applications have been found for these algorithms.

An Ant algorithm consists of 5 structures: a pheromone matrix, an update rule, a probability rule, a heuristic and a local search (see Figure 1.1). A pheromone matrix is how the algorithm learns about its environment and consists of a representation of the relevant relationships involved in the building of a solution. An example in the case

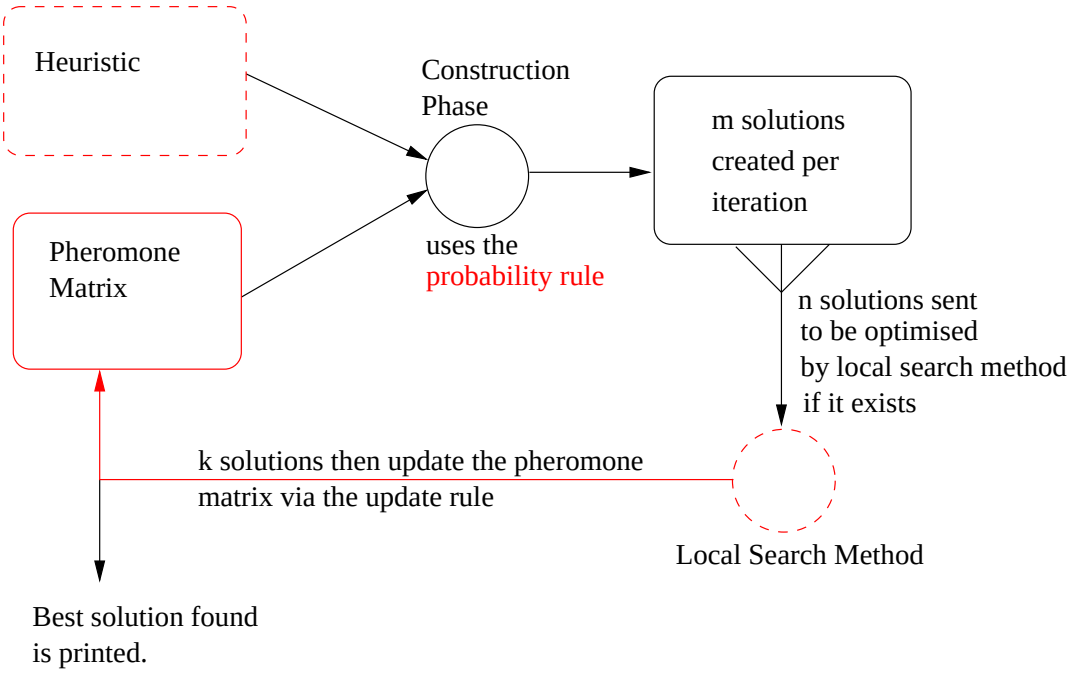


Figure 1.1: A diagram showing the five constituents of an Ant algorithm. (Red indicates one of the discussed components.)

of the Travelling Salesman Problem (see Section 3.3.1) might be a two dimensional matrix whose values indicate the likelihood of city i and city j being adjacent. This is the first source of knowledge for the algorithm and it is an internal source.

The update and probability rules are the input and output mechanisms for the pheromone matrix. The update rule is the method whereby the pheromone matrix from the previous iteration is combined with the information gained from the current iteration. In this way knowledge is acquired by the algorithm. The probability rule is the way in which solutions are constructed and is the way the heuristic and the pheromone matrix are combined to choose the next item for the current partial solution.

The final two structures are the heuristic and the local search, which are both external knowledge sources. Both these structures are specified by the user and may depend on information or relationships not accessible by the algorithm in any other way.

Solutions are created by moving through a graph representation of a problem where each vertex is a possible component of a solution. The probability rule is used to choose the next vertex to visit. The algorithm completes when there are no more vertices to choose from or fulfils a stopping criterion. A number of ants each creates its own solution. These are then evaluated, with a certain number being considered for

use with the local search. The update rule then adds a predetermined set of solutions to the pheromone matrix; most commonly only the best solution found in the run so far is used.

These five structures form the basis of an Ant algorithm. A more detailed overview is given in Chapter 2. However, in the next section the motivation for this thesis is explained. In Section 1.2 more details are given about how problems are represented as graphs and how algorithms traverse these graphs. Following these explanations, the objectives and road-map for the thesis are stated.

1.1 Motivation

Figure 1.2 shows an illustration of the journey that is the reason behind this thesis. In 2002, Ant Colony Optimisation, which is a framework for creating an Ant algorithm, was applied to AI Planning [McCallum and Levine, 2002]. The algorithm was based on Ant System, using both a heuristic and a basic local search. The knowledge introduced by the heuristic was the number of goal states fulfilled if a particular action was added to the plan. At the same time, two local searches were implemented, a depth-first and a breadth-first search to optimise the plan between two particular actions. The results were not particularly encouraging.

In 2003, the Ant System algorithm was applied to University Timetabling and Grid Scheduling, again with limited results, although the problems were, and still are considered very challenging. Progress had been made with these domains, but it was clear that optimising the local search method was much more valuable than working on the Ant algorithm. During this work it was found that the local search method was dominating the solution optimisation process, and that the Ant algorithms were not learning any useful relationships.

[Ritchie, 2003; Ritchie and Levine, 2004, 2003] created a hybrid Ant algorithm with local search for the heterogeneous multi-processor scheduling problem. However, Ritchie later found that better performance could be achieved by swapping the Ant algorithm with a simple shaking algorithm. This is a technique where a solution is jumbled up in a random way to dislodge it from a local optimum. Again this showed that something was not working as expected with the Ant algorithm. Taking into consideration that there are many successful Ant algorithm and local search hybrid sys-

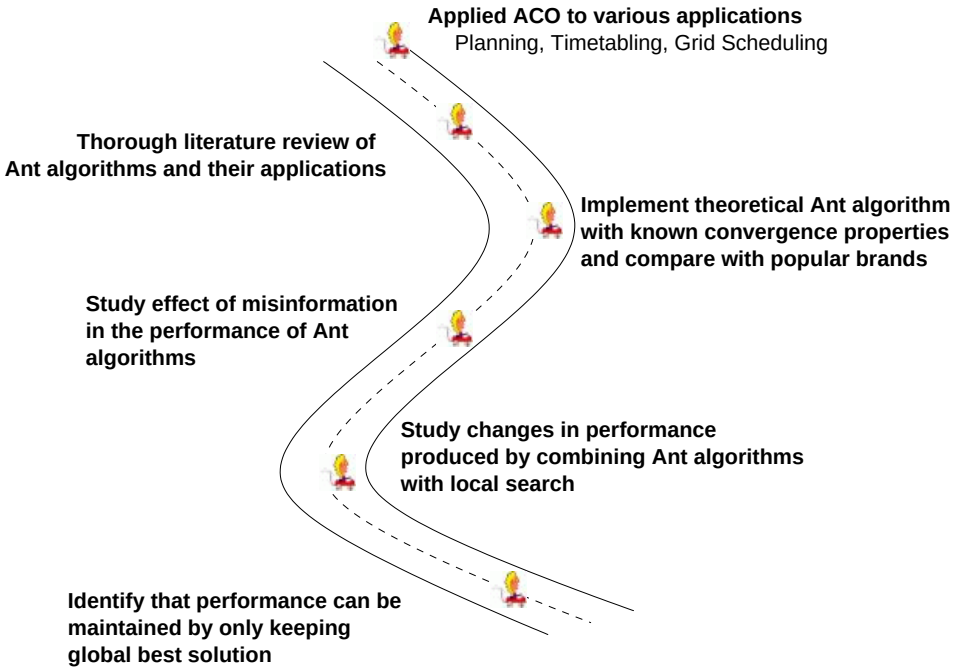


Figure 1.2: Figure illustrating the Thesis Story.

tems that are the best in their field, it is worth asking whether these hybrid algorithms work as their creators think they do. Therefore having tried various applications it was decided the more significant question to ask is "how does local search and heuristic knowledge affect Ant algorithms?"

This thesis looks at algorithms that work on Combinatorial Optimisation Problems represented as graphs, therefore the next section will introduce the reader to the definition of a Combinatorial Optimisation Problem (COP) and why they are hard problems to solve. It will explain how these problems can be represented as graphs, and how algorithms can traverse this graph to construct a solution to the particular problem. It will introduce some technical terms for describing graphs and some definitions to clarify the area of concern. Following this will be a summary of each chapter in Section 1.4.

1.2 The Graphical Nature of Combinatorial Optimisation Problems

A *Combinatorial Optimisation Problem* (COP) is a problem set in a finite domain that has to be either minimised or maximised. In mathematics an optimisation problem

[Boyd and Vandenberghe, 2004] has a general form given in Equation 1.1. In this equation the vector $x = (x_1, x_2, ..., x_n)$ is the variable to be optimised, while the function $f_0 : \Re^n \rightarrow \Re$ is *the objective function*. An objective function maps a solution to a real number vector that represents the quality of the solution in relation to the goals of the problem. The functions $f_i : \Re^n \rightarrow \Re, i = 1, \ldots, m$ are the *constraint functions*, and the constants $b_1, \ldots, b_m$ are the limits, or bounds for the constraints.

$$\begin{array}{ll} \text{minimise} & f_0(x) \\ \text{subject to} & f_i(x) \leq b_i, \quad i = 1, \ldots, m \end{array} \tag{1.1}$$

The vector x^* is called an *optimal* solution to this optimisation problem if it has the smallest objective value among all the vectors that satisfy the constraints: for any z with $f_1(z) \leq b_1, \ldots, f_m(z) \leq b_m$, one has $f_0(z) \geq f_0(x^*)$. Optimisation problems can be split into two classes, *linear* and *non-linear* problems, the distinguishing feature being linear problems satisfy the following constraint:

$$f_i(\alpha x + \beta y) = \alpha f_i(x) + \beta f_i(y) \; \forall x, y \in \Re^n, \alpha, \beta \in \Re \tag{1.2}$$

Implicit in the formal mathematical definition is the concept of two classes of constraints, *hard* constraints, which must be satisfied for a solution to be valid and *soft* constraints, which must only be optimised.

One more classification of optimisation problems is required and that is to define the set of *convex* problems. A convex optimisation problem is one in which the objective and constraint functions are convex, which means they all satisfy the following inequality:

$$\begin{array}{l} f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_j(y) \; \forall x, y \in \Re^n, \alpha, \beta \in \Re \\ \text{with } \alpha + \beta = 1, \alpha \geq 0, \beta \geq 0 \end{array} \tag{1.3}$$

The reason for defining this group is that the Travelling Salesman Problem has been shown to be convex in many cases, although this has only been demonstrated empirically and no proof exists. It is significant because there are many mathematical techniques that can be brought to bear on a problem that has been shown to be convex. It is also believed that the reason why algorithms such as Ant algorithms work very well for some problems and not so well for others is the degree of convexity in the landscape [Gómez and Barán, 2004]. This hierarchy is illustrated in Figure 1.3.

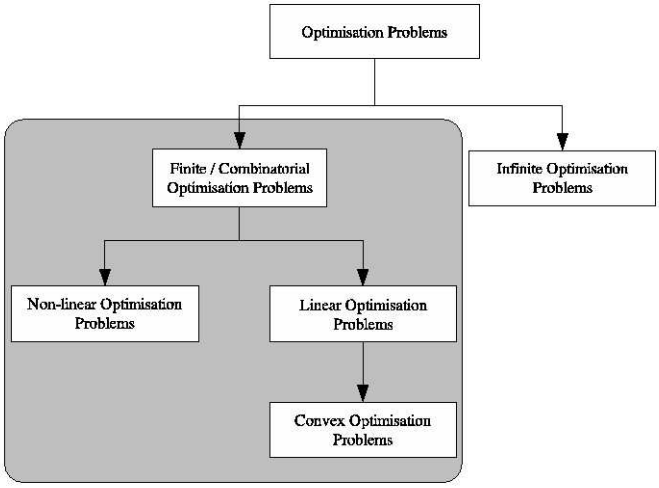


Figure 1.3: Optimisation Problem Hierarchy (The Shaded box indicates the class used in this thesis.)

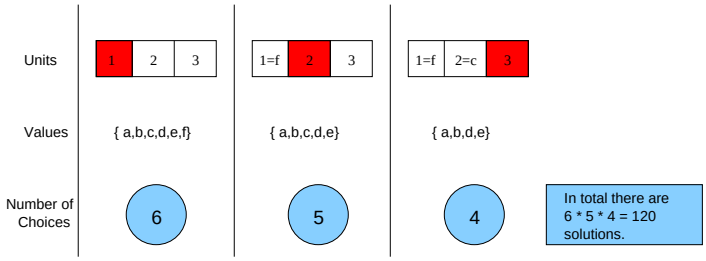


Figure 1.4: An illustration showing how *choice* leads to a combinatorial explosion for a small assignment problem.

Informally, a Combinatorial Optimisation Problem is a problem whose solution is composed of a series of units. Each unit can take a *choice* of values, which can take many forms. Each combination of these unit-value pairs forms a solution which can then be scored with respect to some required features, determined by the author. The problem is solved when this score (called the *objective function*) is maximised, or minimised. At the heart of the problems is the element of choice between values, and it is this choice that gives rise to the combinatorial explosion. Figure 1.4 gives an illustration of this explosion for a small problem of 3 units and 6 values. The total number of combinations is 120, with each unit allowed to choose one of the values.

Due to this combinatorial explosion, Combinatorial Optimisation Problems are very hard to solve, which means that very often these problems are classified in the *Non-Deterministic Polynomial* (NP) class. NP problems are those problems that can only

be verified but not be solved in deterministic polynomial time. Within this set there are two related sets, *NP-Complete* and *NP-Hard*. *NP-Hard* problems are those problems for which all decision problems in NP can be reduced [1] to by a polynomial reduction[2] [Karp, 1972]. If a problem is in the class NP and is NP-Hard then it is called *NP-Complete*. These can be better expressed in the following definitions:

Definition 1.2.1 *For a problem C, C is NP-Hard if* $\forall C' \in NP$, C' *can be modified by a polynomial time reduction to a problem of type C.*

Definition 1.2.2 *For a problem C, C is NP-Complete if C is in NP* and *C is NP-Hard.*

Although the mathematical notation exists, it is often more desirable to reason about these problems in terms of a graph and to be able to apply Graph Theory and its accompanying notation. This graphical representation helps both the understanding of the problem and of the algorithms. The notations and definitions used come from [Boffey, 1982] and therefore will be familiar to those in Operations Research. A complete narrative on Graph Theory is not required but some basic definitions will be related to familiarise the reader with the use of the notation.

An example of a graph representation for a problem is given in Figure 1.5. This representation allows an algorithm to construct a path through the graph, thereby constructing a possible solution to the problem. In the terms described above, each vertex in the graph represents a unit, and the vertices on the arrow end of the arc represent the values. For each vertex (unit) there is a choice of arcs to go down. Thus, it is the constraint of only being able to pick one arc at each stage that leads to the problem's difficulty. Basic algorithms such as Depth First Search and Breadth First Search construct a path by traversing the vertices in a graph in a predetermined order. Both these algorithms are *complete*, referring to the fact that given a finite graph, they will both visit every vertex and construct every possible solution given enough time.

Combinatorial Optimisation Problems can be very hard and also sometimes very large, therefore it is often convenient to approximate the optimal solution rather than to find the mathematical optimum. In Table 1.1 a sample list of both complete and approximate algorithms is given. The disadvantage of the complete algorithms is clearly that if they do not prune the search space effectively they will take a very long time to find a solution. However, their advantage is that the returned solution will be the best so-

[1] A term indicating a transformation by a reduction.

[2] A *reduction* is when there exists an algorithm that can transform a type of problem into another in deterministic polynomial time.

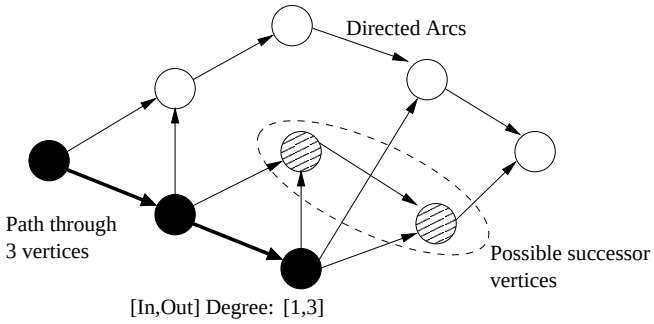


Figure 1.5: Illustration of a directed graph with a path being built up (black nodes), with the successor vertices (shaded nodes).

Complete	**Approximate**
Branch-and-Bound	Simulated Annealing
Cutting Plane Techniques	Genetic Algorithms
Dynamic Programming	Tabu Search
	Iterated Local Search
	Ant Algorithms
	Estimation of Distribution Algorithms
	Cross-Entropy Method

Table 1.1: A sample list of complete and approximate techniques for solving optimisation problems.

lution there can be. For approximate algorithms there is a trade-off between solution quality and time spent searching, and when combined with local search methods the quality of the solutions found is normally acceptable.

A Combinatorial Optimisation Problem, when expressed as a graph, may be directed or undirected, depending on the problem. For each option there is slightly different terminology and therefore, for the sake of clarity, it will be assumed here, unless otherwise stated, that the directed case is being discussed. A summary of the differences in terminology is given in Table 1.2.

A Combinatorial Optimisation Problem, when expressed as a directed graph G, has a set of *vertices* V, sometimes called nodes, and a set of *arcs* A; this is written $G = (V, A)$.

Definition 1.2.3 *A* directed graph *consists of a pair of sets* (V, A) *where*

1. V *is non-empty;*

Directed	Undirected
vertex	vertex
arc (A)	edge (E)
in-degree, out-degree	degree
path	chain
circuit	cycle
loop	loop
elementary	elementary
connected	connected

Table 1.2: Graph terminology for directed and undirected cases. Differences in terminology are highlighted. (Set notation is given in brackets.)

2. *elements of A are directed pairs of, not necessarily distinct, elements of V.*

Elements of V and A will be called vertices *and* arcs *respectively.* $(x,y) \in A$*, or xy as it is normally written, is an arc* from x to y*. Associated with $G = (V,A)$ there is a* successor *mapping $\Gamma : V \longrightarrow V$ where the image of x under Γ, denoted by Γx, is defined by $\Gamma x = \{y | xy \in A\}$. If $y \in \Gamma x$ then y is said to be an (immediate) successor of x and x an (*immediate*)* predecessor *of y.*

This leads to an equivalent second definition in terms of V and the successor mapping Γ.

Definition 1.2.4 *A* directed graph *consists of a pair (V,Γ) where*

1. *V is non-empty,*
2. *Γ is a mapping from V to itself ($\Gamma : V \to V$).*

The reverse of a graph G is $G^{-1} = (V,\Gamma^{-1})$, where $x \in \Gamma^{-1} y$ if and only if $y \in \Gamma x$. If the graph G and its reverse coincide then the graph is *symmetric* and is the corresponding undirected graph. Undirected graphs can be regarded as a special case of directed graphs.

Definition 1.2.5 *An* undirected graph *is a pair of sets (V,E) where*

1. *V is non-empty;*
2. *elements of E, called edges, are undirected pairs of not necessarily distinct elements of V.*

Having defined what a graph is, there is one type of graph that is particularly important to the idea of search and this is the *complete* graph. It is important as Ant algorithms assume that the graph that they traverse is complete. If the problem description suggests an incomplete graph then the algorithm adapts the graph as necessary to make it complete (a process that will be described later).

Definition 1.2.6 *The* complete undirected *graph K_n is the graph on n vertices which contains an edge xy for every pair of vertices x and y. The* complete directed *graph on n vertices is the symmetric directed graph corresponding to K_n.*

As this thesis is looking at an algorithm that constructs its solutions, it is also important to define terminology that relates to solutions in a graph context. To begin with the terms *path* and *chain* are defined that represent partial solutions to the given problem on a directed or undirected graph respectively.

Definition 1.2.7 *A* path *from x_0 to x_s in a directed graph (where x_0 and x_s need not be distinct) is a non-empty sequence of arcs $x_0x_1, x_1x_2, \ldots, x_{s-1}x_s$ and will be denoted by $x_0x_1x_2 \ldots x_s$.*

Definition 1.2.8 *A* chain *between x_0 and x_s in an undirected graph (where x_0 and x_s need not be distinct) is a non-empty sequence of arcs $x_0x_1, x_1x_2, \ldots, x_{s-1}x_s$ and will be denoted by $x_0x_1 \ldots x_s$. The term chain can also be used when referring to directed graphs when directions are ignored.*

A solution for a given problem will normally be represented as a path (or chain). Each arc added to the sequence represents another choice made. Therefore, a path represents a sequence of decisions, which will form a complete solution. The length of this path is normally determined by one or more constraints. When talking about solutions it may be required to characterise the paths in some way. The following two definitions give a couple of useful terms for particular properties.

Definition 1.2.9 *A* circuit *is a closed path (its endpoints coincide), and a* cycle *is a closed chain. A* loop *is a circuit (cycle) containing only one arc (edge).*

In general, Combinatorial Optimisation Problems do not have loops in them. This is because most are permutation and assignment problems and therefore returning to the same node after it had been visited would not be allowed.

Definition 1.2.10 *A path (chain) is* elementary *if no two of its vertices, except possibly x_0 and x_s, coincide.*

An example of an elementary path is a solution for the Travelling Salesman Problem, where the solution consists of all the vertices in the graph starting and ending with x_0, and is therefore a circuit (cycle) - or more precisely a Hamiltonian Circuit. This is explained in more detail in Section 3.3.1.

Definition 1.2.11 *A graph G, whether directed or not, is* connected *if there is a path (chain) between every pair of vertices of G. If G is not connected then it can be split into* components *each of which is a maximal connected graph contained in G.*

Ant algorithms assume that the graphs are all connected and any missing arcs will simply have an infinite, or very large, cost associated with them. This assignment of large costs is also how the algorithms deal with incomplete graphs, as mentioned above. Finally, the two following definitions give a way to communicate how a vertex is connected to the graph in terms of arcs in and out of the vertex.

Definition 1.2.12 *If x is any vertex of a directed graph then the* in-degree $d_i(x)$ *of x is the number of arcs incoming at x and the* out-degree $d_o(x)$ *of x is the number of arcs outgoing from x.*

Definition 1.2.13 *The* degree *d(x) of any vertex x of an undirected graph G is just the number of edges incident at x.*

For most problems, even those represented by directed graphs, the following equality applies: in-degree = out-degree = $|V| - 1$. This is because the graphs are assumed to be complete and connected. As the algorithm moves through the graph it will ignore those arcs that lead to vertices it has already visited, but at the outset this equality can be assumed to hold.

In a computational environment these graph representations are rarely explicitly created within memory. This is because they can grow to become very large and unwieldy for the algorithms to traverse. Therefore the algorithms tend to only consider those parts of the graph that are currently under consideration, in this way the algorithm can work faster and use less memory. In the case of Ant algorithms the parts of the graph under consideration consist of only those vertices that are immediately connected to the current vertex that the algorithm is working on.

This concludes the introduction to Combinatorial Optimisation Problems. The next section introduces the remaining chapters, followed by the contributions given by this thesis.

1.3 Objectives

The following research questions are studied in this thesis:

- Is the Graph-based Ant System representative of both Ant System and the Max-Min Ant System?
- What effect does misleading heuristics have on the three Ant algorithms?
- Does local search alter the properties of the three Ant algorithms?
- What is the reason for any incompatibility between Ant algorithms and local search?

These questions are studied in this order in Chapter 5 to 8. Next is a more detailed layout of the thesis.

1.4 Thesis Road-map

This section outlines the structure of the rest of this thesis, which is split into two parts. The first half deals with the background and details some peripheral problems that occurred on the way to reaching a state ready to tackle the majority of contributions of this thesis. In the second part the contributions to the field of Ant algorithms are made, relying on the base that Part I sets up. This section describes the objectives and purpose behind each chapter in the thesis. In this way it is hoped the reader will be able to navigate to the chapters of their interest.

- *Chapter 2: The Evolution of Ant Algorithms* gives an historical account of the creation of the Ant metaphor from its early biological origins, through its many incarnations, to current applications and theory. Afterward, descriptions are given of the many competitors to Ant algorithms, with the aim of conveying what makes the Ant metaphor distinct. The chapter concludes with a critical look at this work and states some pertinent observations that will be focused on later.
- *Chapter 3: Experimental Methodology* describes the methodology behind the experiments performed. In this chapter the test-suite used to perform the experiments is introduced and the technical details are discussed. Afterwards the problem types and their associated libraries will be described. The chapter ends

with a discussion of statistics and design choices made.

- *Chapter 4: A Local Search Method for Talent Scheduling* introduces a local search method that can be combined with another algorithm to enhance solutions found for the Talent Scheduling Problem. The chapter introduces the technical logic behind the method and shows how this is associated with the pseudo-code. Experiments are performed to show how various forms of this local search compare. The chapter ends with a discussion of these experiments and the final conclusions.

- *Part II: Contributions* summarises the content of the thesis at this stage and sets out in detail the main experimental phase of this thesis. The aim of this summary is to clarify the relationship between the first half of the thesis, which concentrates on background and the general area, and the second half, which relates the majority of the contributions of this thesis.

- *Chapter 5: An Empirical Study of the Graph-based Ant System Model* gives details of the primary theoretical model of how Ant algorithms work. This chapter investigates the properties of the implemented Graph-based Ant System (GBAS) and compares these to those of common algorithms such as Ant System (AS) and the Max-Min Ant System (MMAS). The chapter shows how under certain conditions the model correctly accounts for the behaviour of both Ant System and Max-Min Ant System.

- *Chapter 6: Heuristics as a Source of Knowledge in Ant Algorithms* is an investigation into the behaviour of the three algorithms, introduced in the previous chapter, when misinformation is fed into the algorithms via the heuristic. The chapter discusses five heuristics and demonstrates how different kinds of heuristics have contrasting effects. The chapter ends with a discussion of the conclusions and possible extensions that would allow Ant algorithms to make better use of the heuristic.

- *Chapter 7: An Empirical Investigation of Ant Algorithms with Local Search* combines GBAS, AS and MMAS with local search methods specific to each domain. The chapter compares the performances from Chapter 5 with the new results to show that local search methods change the behaviour of the algorithms. The chapter concludes with a discussion of these changes and discusses the possible ramifications for future research.

- *Chapter 8: Understanding the role of Ant Algorithms when combined with Local Search* takes a step back and tries to identify how the Ant algorithms work when combined with a local search method. The chapter compares the Ant algorithms to both a simple fixed and a variable neighbourhood search. It concludes with a discussion of the possible downfalls of using Ant algorithms with local search and how better problem choices and a different focus for the algorithms could benefit future research.
- *Chapter 9: Conclusions* is the final chapter in the thesis. It brings together the work from the four previous chapters and identifies the major conclusions of this thesis. The chapter discusses what this work means for the community and the future directions it could take.

Chapter 2

The Evolution of Ant Algorithms

Ant algorithms were first introduced in [Colorni et al., 1991; Dorigo et al., 1991a,b] culminating in Marco Dorigo's PhD Thesis [Dorigo, 1992]. This work was inspired by an earlier biological experiment performed with real ant species published in [Goss et al., 1990]. When referring to the biological origins of Ant algorithms the term *Ant metaphor* will be used. It is worth noting that this work is still of interest in the field of biology and more recent work such as [Vittori et al., 2004] is providing new ways of thinking about similar problems.

However, it is not just ants that exhibit interesting behaviours. Social animals of all kinds show a wide variety of communication techniques and behaviours. A number of these behaviours have been studied such as Division of Labour, Brood Sorting and Co-operative transport, an application of which is helping roboticists to design distributed control algorithms [Bonabeau et al., 2000]. Bees, studied in [Wedde et al., 2004], are an example of another social insect that has enabled researchers to solve problems in a variety of innovative ways. However, the majority of research has concentrated primarily on applying the metaphor of ant behaviour to various problems.

In this chapter the principal Ant algorithms will be described, alongside some of their applications. Although this chapter contains a thorough review of the field, the algorithms used later in this thesis are Ant System in Section 2.2.1, MMAS in Section 2.2.6, and the Graph-based Ant System in Section 2.2.10.

In Section 2.1 the original biological experiments are examined. In Section 2.2 the general framework for all the algorithms in the form of the Ant Colony Optimisation Metaheuristic is given followed by the first influential implementation, Ant System in

Section 2.2.1. Throughout the following research there are a number of trends that have developed:

- *Inspiration from another field*: Bringing in ideas from other algorithms and research areas is a valid and interesting way of advancing research. Fields that have been looked at are: Reinforcement Learning (Ant-Q), Binary Integer Programming (Hypercube Framework) and Beam Search from Operations Research.
- *Parameterisation*: In any algorithm that involves a multitude of parameters research is required to know how the algorithm reacts to various value sets. Research has been done on a number of algorithms' parameters such as Ant System, the Max-Min Ant System, the Fast Ant System and Approximated Nondeterministic Tree Search.
- *Theory*: With any initially empirical field there is interest in a theoretical background that can explain the empirical results. A small amount of this work has been done in the field including the Graph-Based Ant System, Simple ACO and some work with the Max-Min Ant System.
- *Applications*: These algorithms have been successfully applied to many domains, making up the bulk of the research. Along with direct implementations, research has been done to transfer the algorithms to continuous domains, parallel and multiple colony architectures.

In Section 2.3 a discussion of the various ways in which the model's parameters are selected will complete the first half of the chapter. During this review particular runs of algorithms will not be compared as it is difficult to assess the quality of the experiments. It is evident from papers that a range of values for parameters have been experimented with, but the optimisation of parameters of compared algorithms has not been documented carefully, therefore any comparison made may not be valid. Alternative search algorithms are often used as comparisons and these are described in Section 2.4 to highlight the distinct characteristics of the Ant algorithms.

2.1 The Biological Inspiration

In this section a more detailed look is given to the paper by [Goss et al., 1990], which was cited as the origin of Dorigo's work. This paper detailed experiments that resulted

in two important findings using social insects. The first showed that when faced with a choice between two routes the colony would over time converge on the shortest route. The second discovery was that the probability of converging on the shortest branch increased with the difference in length between the two branches. This selection behaviour was explained in terms of positive feedback (*autocatalysis*) and differential path length; together these resulted in an indirect method of communication which was labelled *stigmergy*.

The original experiment placed Argentine ants (*Iridomyrmex humilis*) at the start of a bridge with two equal paths (as shown in the inset of the top image in Figure 2.1) and got them to select a route over time. A number of interesting traits were demonstrated. Firstly, the probability of an ant choosing what became the collectively selected branch increased rapidly and non-linearly with the number of ants that had previously passed on the bridge. Second, each ant was expected to lay an average amount of pheromone when leaving or returning to the nest, which made the ant's choice not only a function of time but also of the cumulative number of ants that had passed the measuring point.

This exploration behaviour was modelled using the two equations below. After i ants had crossed the bridge there remained i pheromone units on the bridge, of which A_i were on branch A and B_i on branch B. Equation 2.1 gives the probability of an ant choosing branch A or B, which is dependent on A_i and B_i. Having chosen, the ant then added to the pheromone on the chosen branch, as calculated in Equation 2.2. In this second equation, δ is a stochastic variable taking the value 0 or 1 with probability $p(A)$ and $p(B)$ respectively. The values of 2 for the power and 20 for the coefficient were fitted to the results of these experiments.

$$p(A) = \frac{(20+A_i)^2}{((20+A_i)^2+(20+B_i)^2)} = 1 - p(B) \tag{2.1}$$

$$A_{i+1} = A_i + \delta \quad B_{i+1} = B_i + (1-\delta) \quad (A_i + B_i = i) \tag{2.2}$$

The second experiment consisted of two unequal paths as shown in Figure 2.2. This experiment demonstrated that the selection of the shortest branch increased with the difference between the two branches. An interesting finding was that once a trail had been started on the longer branch, when the shorter branch was introduced the ants were incapable of switching back.

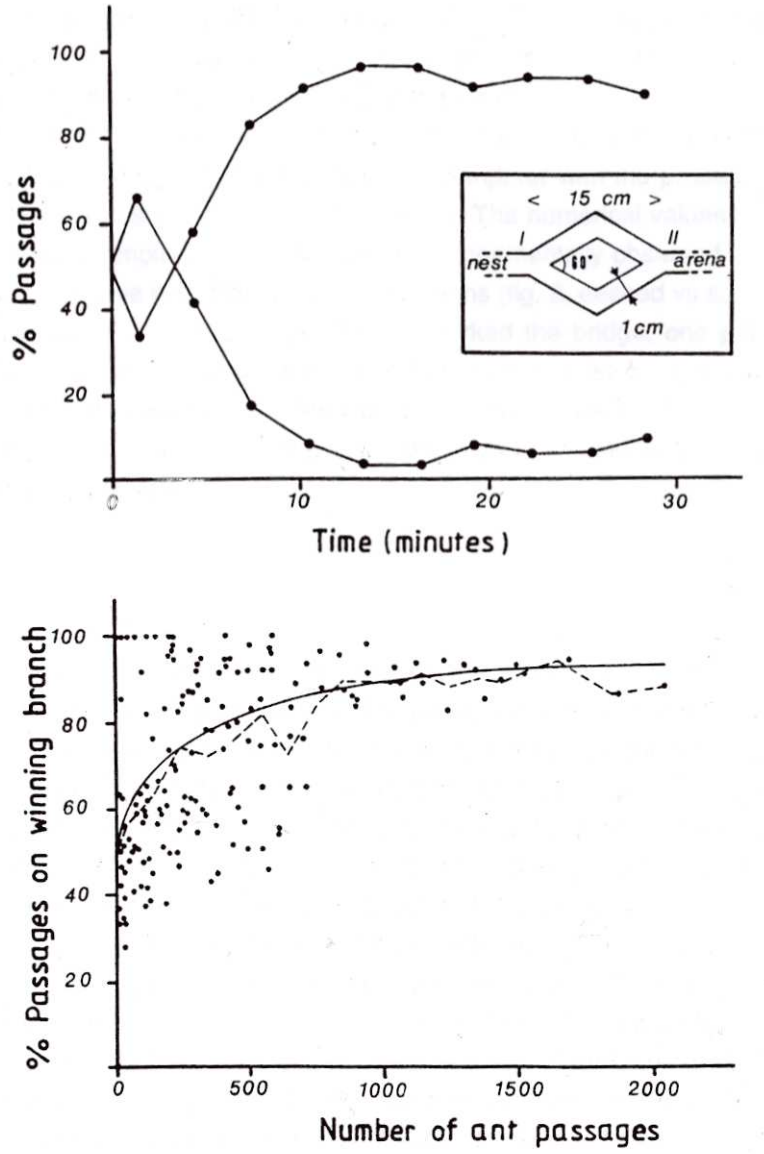


Figure 2.1: Taken from [Goss et al., 1990]. The top graph shows the percentage of workers per 3-minute period passing on the two branches of the bridge (insert). The bottom graph the percentage of workers passing on the collectively selected branch of the bridge. The dots are the results of 20 experiments measuring 3-minute periods for 30 minutes, the dotted line is the average of these experiments and finally the bold line is the average of 200 Monte-Carlo simulations based on Equations 2.1 and 2.2.

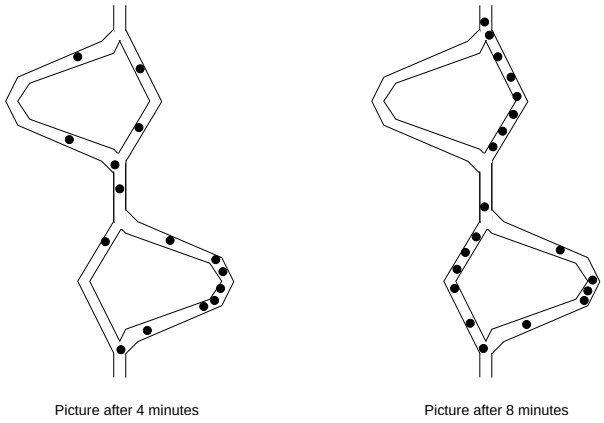


Figure 2.2: Reconstruction of a colony of *Iridomyrmex humilis* selecting the short branches on both modules of the bridge (original photos were taken 4 and 8 minutes after the bridge was placed). [Goss et al., 1990] Black dots are representative of the ants in the original photos.

In the same paper a second set of experiments was undertaken using the *Lasius niger* species of ant. The objective was to show how the mechanism demonstrated in the previous experiment would enable a colony to choose the richer of two food sources. In a previous piece of research it had been found that this type of ant species searched for food in a number of directions centred around the colony. Therefore an experiment was devised that placed the food sources at varying locations, thus paths were only reinforced if food was found in that particular direction. It was found that as food became more scarce at one particular outlet fewer ants continued to visit that location; this was due to pheromone evaporation causing the path to disappear. Depending on the number of foragers or food sources, the pheromone trail rotated around the colony at different rates. The same mechanism could result in random foraging (*exploration*) or a fixed trail (*exploitation*).

The model in Equation 2.3 used in this experiment is similar to the last experiment. However this equation took into account the arrival and disappearance of food sources containing seeds (an interesting analogy to dynamic optimisation problems). The circular foraging area around the colony was split into b sectors. ϕ_i seeds arrive per time unit in each sector, each of which contains S_i seeds. A fraction of the seeds, r, disappear per time unit to simulate competition, decay and other extraneous factors.

A trail was assumed to lead to each sector with C units of pheromone along it, of which a fraction, e, evaporated per unit time. N foragers left the nest per unit time. A

fraction, f_i, would choose sector i according to the multi-choice probability equivalent to Equation 2.1. A smaller fraction of this, an amount q, diffused into each of the adjacent sectors.

The number of seeds that were found in a sector was given by $g \cdot \frac{F_i S_i}{(a+S_i)}$ where F_i was the number of foragers in sector i, a was a constant parameter and g was an experimental constant. At the end of each step, all the foragers returned to the nest and those that had found a seed added one unit of pheromone to the corresponding trail, while those that had not found any food did not lay any pheromone.

$$\begin{array}{lll} \frac{dS_i}{dt} = & \phi_i - g \cdot \frac{F_i S_i}{(a+S_i)} - r \cdot S_i & (i = 1, \ldots, b) \\ \frac{dC_i}{dt} = & g \cdot \frac{F_i S_i}{(a+S_i)} - e \cdot C_i & \\ F_i = & N((1-2q) \cdot f_i + q \cdot f_{i+1} + q \cdot f_{i-1}) & \\ f_i = & \frac{(20+C_i)^2}{\sum (20+C_j)^2} & \sum f_i = 1 \end{array} \tag{2.3}$$

At the end of the paper a number of interesting questions were raised by the audience at the conference. The first answer related the point that the decay rate played an important part in the model and that if decay is very rapid one needed large numbers of ants active together. This suggested a relationship between the numbers of ants and the decay factor. A later question asked what governed the diameter of the search circle that the ants seemed to use. The suggestion was that the diameter would increase with each sweep of the circle, perhaps alluding to a novel way to handle variable-length solutions to problems. Finally there was also the suggestion of decay being proportional to the quality of the source. The idea behind this was that if a number of ants came back empty after a long period of fruitful search the decay factor could be increased to hasten a change in search direction.

Capturing this biological problem-solving architecture was the aim of the original Ant algorithms. Having looked at the biology, the computational model is now described. The computational model tries to emulate this behaviour as closely as possible. The model consists of a graph, as defined in Chapter 1.2, that acts as the maze for the ants. Each ant, or agent, can be processed in parallel or in serial, and its purpose is to perform a probabilistic walk of the graph, known as a *trail*. At some point, either after each step or at the end of the walk, the ant will imprint its solution in a memory, known as the *Pheromone Matrix*. Each value in the matrix is known as a *trail intensity*. The solution is saved in the matrix using a *trail update rule* that updates the values in the

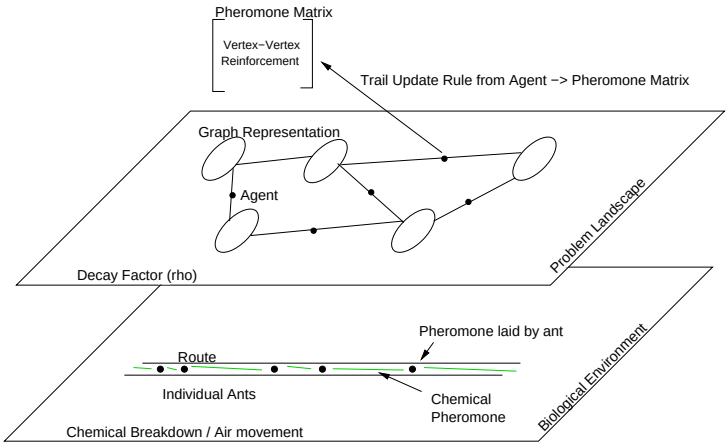


Figure 2.3: A visual representation of the computational model that has been inspired by its biological counterpart.

matrix with a quantity of pheromone, which may be related to the objective value of the solution. Another function of the update rule is to reduce those values not contained in any of the solutions in the update, this reduction is known as *decay*. This abstraction is illustrated in Figure 2.3.

To summarise this section a brief discussion of the similarities and differences between the biological and artificial ants is given (based on an extract from [Dorigo and Di Caro, 1999a]).

- **Colony of cooperating individuals** : In biology ants live in colonies and this is modelled directly into the algorithms with each ant being part of a population, or colony, of concurrent and asynchronous entities that cooperate on a global scale to achieve a set task.

- **Pheromone trail and stigmergy** : Real ants modify their environment to achieve communication between individuals and in the same way artificial ants use a memory mechanism to store some numeric information about the states they visit achieving the same indirect communication. As with real ants this information changes the way that the environment is perceived by each individual. In biology this chemical information degrades and in a similar way the numeric information is reduced by the decay factor.

- **Shortest path searching and local moves** : Both types of ants walk to adjacent areas of the landscape and cannot hop between different areas. In a computational sense what counts as adjacent is problem specific.

- **Stochastic and myopic state transition policy** : Both ants rely on two sources of information, local environment modifications in the form of pheromone and also a priori information in terms of local terrain, or in the artificial sense a lookahead function (heuristic).

These four aspects are certainly modelled in some respect but this does not take into account certain criticisms of the application of the metaphor. The following describes some criticisms that are voiced about the relevance of the biological model to a computational one.

- **Discrete versus Continuous** : The problem landscapes in biology are continuous and dynamic, whereas the problems normally solved by these algorithms in research papers are discrete and static.

- **Simplicity of problem** : The problem that the ants are solving in the papers described earlier are very simple. The methods the ants use may not necessary be the best or even relevant to more difficult problems in a computational setting.

- **Applicability of communication strategy** : The biological communication mechanism may not be the best way to allow computational agents to communicate. While numerical information is stored in a memory, from a practical point of view there is no reason why the memory must be numeric.

- **Distributed nature** : In general all the ants lay pheromone and there are many thousands of them. In a computational setting this is rarely the case, normally one or two candidates, selected from a small sample, are impressed on the memory.

- **Relevance of the environment** : In the natural world the environment is constantly in flux and therefore the decay of the pheromone level is necessary. In a computational setting a decay factor may not be relevant, thus it is possible that there exists a better mechanism for achieving replacement of old information.

The discussion above describes a number of arguments used by both sides in debating the relevance of the use of a biological metaphor once a computational model has been derived. In the remaining half of the chapter, the various versions of Ant algorithms are described with the intent of illustrating the conflicts and biases that have moulded them, followed by a detailed look at how parameters are selected and how they affect the algorithms' performance.

2.2 Ant Colony Optimisation Metaheuristic

The Ant Colony Optimisation (ACO) Metaheuristic brings together a set of algorithms that have taken inspiration from the biology described in the previous section [Dorigo and Di Caro, 1999a,b]. This framework was not part of the original studies but was created by the main proponents of these methods to bring together a number of similar algorithms into a single field. It embodies the architecture of all the following algorithms therefore this will be the starting point for this review.

The term *metaheuristic* was first coined by Glover in 1986 in the first publication of Tabu Search. The term refers to "a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality" [Glover and Laguna, 1997]. The Ant Colony Optimisation Metaheuristic can be viewed as a framework for guiding other local search algorithms. There are other metaheuristics that do a similar and equally effective job and these will be discussed later in this chapter. The framework evolved from a number of algorithms, an illustration of which can be seen in Figure 2.4. This diagram shows the evolution of the framework from the first algorithms to current implementations and theoretical models. Although the diagram is not exhaustive, it does show the main algorithms.

The ACO framework is outlined in the three procedures:

- `ACO_Meta_Heuristic`,
- `ants_generation_and_activity`,
- and `new_active_ant`.

It is designed to be flexible allowing it to incorporate parallel and serial architectures, as well as other ant-like agent algorithms. As such it is useful more for classification, rather than practical purposes. The most common notations used in the framework are given in Table 2.1, and the others will be introduced as they are used in a particular algorithm.

The first procedure, `ACO_Meta_Heuristic`, is the outermost body of the framework and raises a number of design decisions. All the algorithms employ the same mechanisms for termination criteria and these are given in the following list.

- A maximum number of cycles have been processed.

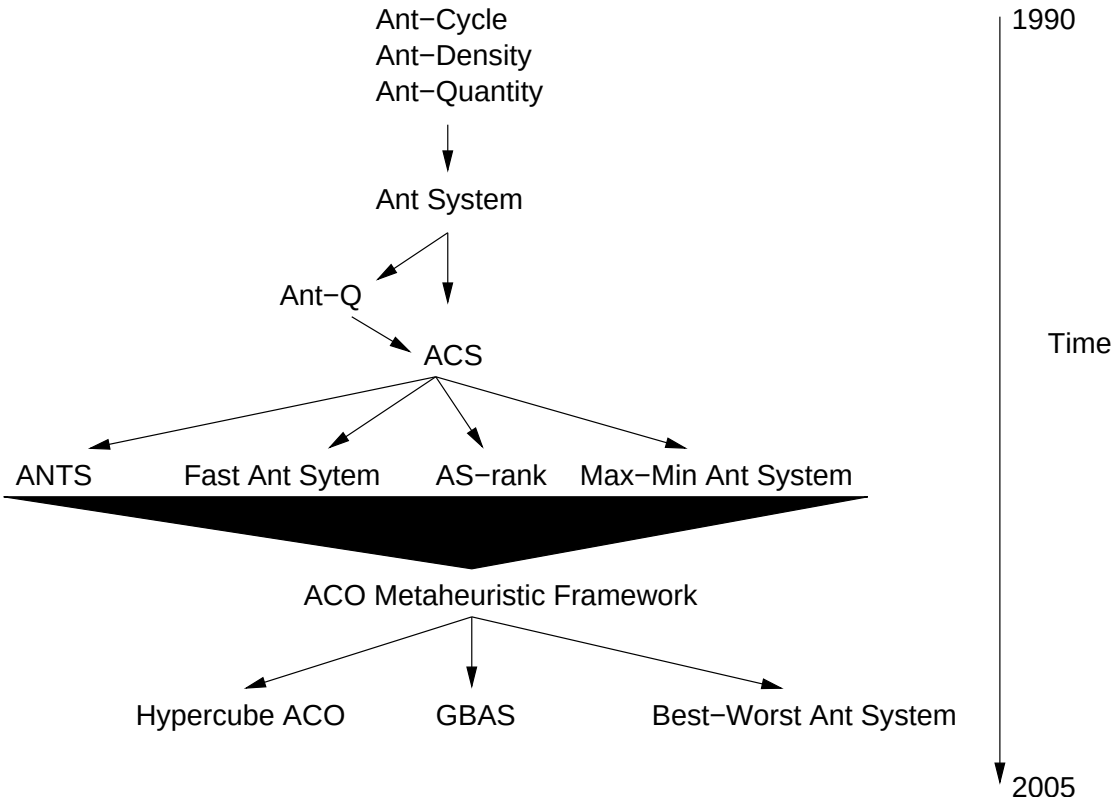


Figure 2.4: A family tree of Ant algorithms, displaying how the various algorithms fit together. (Not a complete list of Ant algorithms, only the most prominent algorithms are featured.)

Notation	Description
t	One iteration of the while-loop in the Procedure 'ACO_Meta_Heuristic'.
m	Number of ants per iteration.
$\phi_{ij}^{k}(t)$	The value of a variable or function ϕ at iteration t for solution k on edge (i, j). This is a general notation that the variables will be introduced with. If k is missing from the notation it should be assumed the variable or function applies to all ants.

Table 2.1: Common Notation

- The algorithm has run for a maximum amount of time.
- A solution has been found with the a priori global optimum.
- The algorithm has stagnated and not found a better solution than its best so far for a determined number of cycles.

The criteria are familiar as there are similar rules in other approximate algorithms such as Genetic Algorithms or Simulated Annealing. One pass through the *while* loop of this procedure is known as a *cycle*, or iteration. These two terms will be used interchangeably.

The procedure, `ants_generation_and_activity`, generates a set of solutions. This can be done in parallel and is usually parameterised by a variable m, the number of ants to create per cycle. As part of this procedure `new_active_ant` is called which processes each new ant, describing how a solution is constructed and the update of the pheromone matrix. For this procedure to be efficient, the correct graph structure must be identified for the problem and the best reinforcement policy to apply must be chosen.

A function called `pheromone_evaporation` reduces the pheromone intensities in a pre-determined manner. This is followed by the function `daemon_actions`, which represents the situation where some extra commands need to be processed on a global scale, for instance recording all ant solutions created or perform some local search method on a particular individual.

There are many implementations of this framework and it is important to view each algorithm with its motivating domain to gain understanding about the differences that define them as separate algorithms.

Procedure ACO_Meta_Heuristic

while *termination_criterion_not_satisfied* **do**

begin schedule_activities

ants_generation_and_activity();

pheromone_evaporation();

daemon_actions(); *optional*

end schedule_activities

endw

Procedure ants_generation_and_activity

while *available_resources* **do**

schedule_the_creation_of_a_new_ant();

new_active_ant();

endw

Procedure new_active_ant

initialise_ant();

$M \longleftarrow$ update_ant_memory();

while *current_state* $\neq$ *target_state* **do**

$A \longleftarrow$ read_local_ant-routing_table();

$P \longleftarrow$ compute_transition_probabilities(A,M,problem_constraints);

next_state $\longleftarrow$ apply_ant_decision_policy(P, problem_constraints);

move_to_next_state(next_state);

if *online_step-by-step_pheromone_update* **then**

deposit_pheromone_on_the_visited_arc();

update_ant-routing_table();

endif

$M \longleftarrow$ update_internal_state();

endw

if *online_delayed_pheromone_update* **then**

evaluate_solution();

deposit_pheromone_on_all_visited_arcs();

update_ant-routing_table();

endif

die();

2.2.1 Ant System

Published after the initial thesis by Dorigo, the Ant System (*AS*) is the first coherent and most prominent Ant algorithm. This algorithm expounded a number of characteristics, *positive feedback*, a *distributed architecture* and a *solution construction procedure*. It is based on three initial attempts at defining the algorithm, Ant-Density, Ant-Quantity [Dorigo et al., 1991a] and Ant-Cycle [Colorni et al., 1992].

The main components of the algorithm are: *a pheromone matrix* (τ), which is the memory of the algorithm, allowing indirect communication between ants; *a trail update rule* (Equation 2.4), which is an equation defining how new solutions are integrated into the pheromone matrix; ρ, a parameter that reduces the pheromone on unused edges, sometimes referred to as the learning rate or *pheromone decay*; and $p_{ij}^k(t)$, a *probability rule* (Equation 2.5) that at each junction from a vertex i to some vertex $j \in J$ determines what edge is chosen by the k-th ant. The set J is the neighbourhood of possible successor vertices from vertex i.

$$\begin{aligned} &\tau_{ij}(t+1) \longleftarrow \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t,t+1) \\ &\text{where } \Delta\tau_{ij}(t,t+1) \longleftarrow \textstyle\sum_{k=1}^{m} \Delta\tau_{ij}^k(t,t+1) \end{aligned} \tag{2.4}$$

$\Delta\tau_{ij}^k(t,t+1)$ is the quantity of pheromone per unit of length of trail laid on the edge (i,j) by the k-th ant between time t and $t+1$.

$$p_{ij}^k(t) \longleftarrow \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{r \in \text{ tabu}_k} [\tau_{ir}(t)]^\alpha [\eta_{ir}]^\beta} & j \in \text{tabu}_k \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

The probability rule combines two sources of information; the pheromone trail intensity (τ_{ij}) and the user-specified heuristic (η_{ij}), also called the *visibility*. The heuristic is an estimate of future success which is dependent upon how deep into the search tree it looks. The two quantities are weighted via two parameters α and β. The set of vertices available for the ant to move to is restricted by a Tabu list ($\text{tabu}_k \subseteq J$) that removes vertices already in the partial solution.

The three original algorithms were applied to the Travelling Salesman Problem and differed in the amount of pheromone laid and the timing of the trail update. Ant-Density used a constant update after every step an ant took, while Ant-Quantity used an amount proportional to the distance between cities i and j ($\frac{Q}{c_{ij}}$, where Q was an

Parameter	Value Range	Chosen value
α	$\{0.3, 0.5, 0.9\}$	0.5
ρ	$\{0.7, 0.9, 0.95, 0.99\}$	0.9

Table 2.2: Table showing the parameters sets used in Ant System applied to the Quadratic Assignment Problem [Maniezzo and Colorni, 1999b].

arbitrary parameter and c_{ij} was the cost of moving from city i to city j). Ant-Cycle was the first to perform the trail update at the end of the construction process and was updated with a value proportional to the length of the tour, $\frac{Q}{L^k}$, where L^k was the length of the k-th tour. All these algorithms had a complexity of $O(NC \cdot n^3)$, where NC was the number of cycles, n was the number of cities and $O(\cdot)$ is an asymptotic upper bound for the magnitude of a function in terms of another, usually simpler, function [Cormen et al., 2001]. Ant System implements the Ant-Cycle method of updating at the end of the `ants_generation_and_activity` function in Procedure `ACO_Meta_Heuristic`.

In [Maniezzo and Colorni, 1999a,b] the Ant System was applied to the Quadratic Assignment Problem. In the later version the algorithm used a slightly modified probability equation, shown in Equation 2.6.

$$p_{ij}^k(t) = \begin{cases} \frac{\alpha \cdot \tau_{ij}(t-1) + (1-\alpha) \cdot \eta_{ij}}{\sum_{r \in tabu_k} (\alpha \cdot \tau_{ir}(t) + (1-\alpha) \cdot \eta_{ir})} & \text{if } j \in tabu_k \\ 0 & otherwise \end{cases} \tag{2.6}$$

In the experiments that were performed a range of values were tried for each parameter, given in Table 2.2. Although the higher value of α led to stagnation, no mention was given of the lower value - presumably it failed to produce the required quality of solution. The lower values of ρ for the pheromone update rule reduced the algorithm's efficiency, meaning that the algorithm took longer to find good solutions. Finally it was found that the number of ants did not have a decisive influence on the overall performance. These experiments were done with the addition of a local search method described in the paper.

The Ant System algorithm combined with local search found solutions of comparable quality to, and slightly faster than, the GRASP algorithm from 1994 [Li et al., 1994b]. This is an interesting result as GRASP is very similar to the ACO Metaheuristic. Procedure `GRASP` gives the GRASP algorithm from [Resende and Ribeiro, 2003] and one can immediately see the similarity. The differences lie in the way that the GRASP

algorithm generates new solutions. The algorithm for the construction process is in Procedure `Greedy_Randomised_Construction` and new solutions are generated via a threshold value and random selection. Back in 1994 the threshold was set by the researcher, but subsequent work has made a reactive version of the algorithm that tunes this parameter, thus making the algorithms even more similar. The most significant difference between the two algorithms is in their use of memory. GRASP records the very best solution created so far but has no long term memory. This means that it does not have the ability to remember clues that it used a few cycles before. Maniezzo and Colorni concluded that the fact that the Ant System outperformed GRASP in most cases showed how important the memory capability achieved by the use of the pheromone matrix was.

Procedure `GRASP`(*Max_Iterations,Seed*)
Read_Input();
for $k = 1,\ldots,$*Max_Iterations* **do**
    Solution $\longleftarrow$ `Greedy_Randomised_Construction`(Seed);
    Solution $\longleftarrow$ Local_Search(Solution);
    Update_Solution(Solution, Best_Solution);
endfor
return *Best_Solution*;

Procedure `Greedy_Randomised_Construction`(*Seed*)
Solution $\longleftarrow \emptyset$;
Evaluate the incremental costs of candidate elements with respect to threshold α;
while *Solution is not complete* **do**
    Build the restricted candidate list (RCL);
    Select element s from the RCL at random;
    Solution $\longleftarrow$ Solution $\cup \{s\}$;
    Re-evaluate the incremental costs;
endw
return *Solution*;

Ant Systems have also been applied to graph colouring in [Costa and Hertz, 1997]. This paper gave a generalised algorithm for Assignment Type Problems (ATPs) which is given in Figure 2.10. The distinctive quality about this algorithm was that not only did it have a memory for assigning items to resources, but it also had a memory for the ordering of the items. Many algorithms before and since this paper have tended

Order	Description
Static	
Random	Vertices are ordered randomly
Largest First	Vertices are ordered so that their degrees $deg_v(v)$ form a non-increasing sequence.
Smallest First	The order $v_1, v_2, \ldots, v_n$ is such that each vertex v_i has smallest degree in the subgraph induced by vertices $v_1, \ldots, v_i$.
Dynamic	
DSaturation	Let A be the set of all vertices not yet been coloured. Vertices are ordered by choosing at each step the vertex $v \in A$ with a minimum degree of saturation $dsat(v)$. If v is not unique preference is given to the vertex v for which $deg_A(v)$ is minimum.
Recursive Largest First (RLF)	Classes of colours are built sequentially. Once a vertex $v \in A$ with maximum degree $deg_A(v)$ has been selected, the current stable set is augmented by inserting, as long as possible, the vertex $v \in AB$ with maximum degree $deg_B(v)$. The set B contains every uncoloured vertex, which can no longer be included in the stable set under construction.

Table 2.3: The various node orderings used in [Costa and Hertz, 1997] where Ant System was applied to the Graph Colouring problem. RLF was reported as the best ordering.

to assign resources to items in a linear manner. In this paper two dynamic orderings were used as the heuristic for the probability of assigning an item at a particular step; these orderings can be seen in Table 2.3. The best variation out of the test set was RLF, although there was a trade-off between run-time and the number of colours used. The paper also investigated various combinations of α and β ranging from $[0,0]$ to $[4,4]$. Generally it achieved better results when $\beta > \alpha$.

To summarise, there were two main outcomes of this paper. It demonstrated the flexibility of the Ant algorithm to cope with various assignment problems, and it also showed that the size of the colony was critical on a non-parallel machine to make a reasonable trade-off between solution quality and runtime. The authors noted that above a certain number of ants the extra information no longer helped the algorithm significantly in this trade-off.

Figure 2.10: Generalised AS for Assignment Type Problems [Costa and Hertz, 1997]

1 $p_{it}(k,i)$ is the pheromone matrix for item i at step k;
2 $p_{re}(k,i,j)$ is the pheromone matrix for assigning location i to resource j at step k;
3 $f^o \leftarrow \infty$ (best value reached so far);
4 J_i is the set of admissible resources;
5 ncycles $\leftarrow 0$ (cycle counter);
6 best_cycle $\leftarrow 0$ (cycle at which the best solution has been found);
7 $\tau_1(s[k-1],i) \leftarrow 1; \tau_2(s[k-1],i,j) = 1; \forall i = 1,\ldots,n; \forall j \in J_i; \forall s[k-1], k = 1,\ldots,n;$
8 **while** *stopping criteria = false* **do**
9 ncycles $\leftarrow$ ncycles $+ 1$;
10 **for** $a = 1$ *to* m **do**
11 **for** $k = 1$ *to* n **do**
12 choose an item $i \in \{1,\ldots,n\}\ \{o(1),\ldots,o(k-1)\}$ with probability $p_{it}(k,i)$;
13 choose a feasible resource j for item i with probability $p_{re}(k,i,j)$;
14 assign resource j to item i; $x_{ij} = 1; x_{il} = 0; \forall 1 \neq j; o(k) = i;$
15 **endfor**
16 compute the cost $f(s_a)$ of solution $s_a = (x_{11},\ldots,x_{nm})$;
17 **endfor**
18 $P = \{s_1, s_2, \ldots, s_m\}$;
19 $\tilde{s} = \arg\min\{f(s) | s \in P\}$;
20 **if** $f(\tilde{s}) < f^o$ **then**
21 $s^o = \tilde{s}$; $f^o = f(s^o)$; best_cycle = ncycles;
22 **endif**
23 update trails τ_1 and τ_2;
24 **endw**

In [Eyckelhof and Snoek, 2002] the Ant System was applied to the Traffic-Jam Dynamic TSP problem, where the cost of a particular segment of the current optimal journey was altered as if a traffic jam had formed along that route. To accommodate this problem a number of modifications were made. The main change was that the pheromone values were *shaken* after a change to the problem. This kept the relations between pheromone values the same, while the pheromone intensity of non-optimal solutions was raised so that exploration would be increased; the formula for this is given in Equation 2.7.

$$\tau_{ij} = \tau_{min} \cdot \left(1 + \log\left(\frac{\tau_{ij}}{\tau_{min}}\right)\right) \quad \text{where } \tau_{min} \text{ is the minimum amount of pheromone} \tag{2.7}$$

The paper states that one problem with this equation was its global scope, meaning it was applied to the pheromone values of all edges whereas it was probable that one could only apply this in the vicinity of the altered path. For this reason a *local shaking* procedure (`localshaking`) was also defined. It was applied to roads within a distance of p·MaxDist of the affected edge, where *MaxDist* was the maximum distance between any two cities in the original problem without traffic jams and p was in the range 0 to 1.

Procedure `localshaking`

if *distance between cities* (a,b) *changed* **then**
  for *every road* ij **do**
    if $(d_{ai} < p \cdot MaxDist) \vee (d_{bi} < p \cdot MaxDist) \vee$
    $(d_{aj} < p \cdot MaxDist) \vee (d_{bj} < p \cdot MaxDist)$ **then**
      $\tau_{ij} = \tau_0 \cdot \left(1 + \log\left(\frac{\tau_{ij}}{t_0}\right)\right)$;
    endif
  endfor
endif

Another significant point about this experiment was that it used a minimum but not a maximum amount of pheromone. This was to make sure that the pheromone intensity never reduced to 0. This is the only paper, of the ones reviewed, based on Ant System that implements a minimum pheromone value.

HAS-QAP was based on the Ant System and was introduced in [Gambardella et al., 1999b]. In this version the solution construction phase was altered so that instead of

creating a completely new solution, the current best solution was modified by performing a number of swaps. The swaps were performed by choosing the first item i at random. The second item was then chosen from the remainder $n-1$ items. Exploitation was achieved with probability $1-q$, where q is a parameter in the range $[0,1]$, by maximising the value of $\tau_{j,p(i)} + \tau_{i,p(k)}$, where i and j are locations, and $p(i)$ and $p(j)$ are resources placed at locations i and j in the context of a QAP problem (see 3.3.2). Alternatively, exploration was achieved probabilistically using Equation 2.8.

$$p_{ij}^{k}(t) = \frac{\tau_{ip(j)}(t) + \tau_{jp(j)}(t)}{\sum_{l \neq i}(\tau_{ip(l)}(t) + \tau_{lp(i)}(t))} \tag{2.8}$$

The parameter q was set to 0.9 and the number of ants per iteration was set to 10. The algorithm was very similar to an Iterated Local Search and was shown to perform competitively in comparison to a Simulated Annealing algorithm, a Tabu Search and some hybrid methods based on Genetic Algorithms.

2.2.2 Ant-Q

The first major adaptation to Ant System was published in [Gambardella and Dorigo, 1995], with a reworking in [Taillard and Gambardella, 1997b]. It was a fusion of Ant System and Reinforcement Learning. The underlying reason for trying this combination was their similarity in purpose. Both research fields try to learn about their current environment from their experiences of moving around that environment. Reinforcement Learning also brought some potentially useful concepts such as *horizons* and the *Bellman Optimality Equations*. Horizons are a mathematical way of discounting older evidence in a gradual way, thereby giving more weight to the most recent information. The Bellman Optimality Equations allowed the calculation of the policy that gives the best expected return for expected rewards. Ant-Q has been applied to both Quadratic Assignment Problems and Travelling Salesman Problems [Dorigo and Gambardella, 1996].

The algorithm replaced the pheromone matrix with another set of values known as *Ant-Q-values*, AQ_{ij}. These values were calculated in a similar way as Q-learning Q-values in Reinforcement Learning and represent the usefulness of a particular move. These AQ-values were updated by Equation 2.9, which is similar to the trail update rule.

$$AQ(r,s) \longleftarrow (1-\rho)\cdot AQ(r,s) + \rho\cdot(\Delta AQ(r,s) + \gamma\cdot \max_{z\in J_k} AQ(s,z))$$
$$\text{where} \quad J_k \text{ is the list of cities still not visited by ant } k \qquad (2.9)$$
$$\gamma \text{ is the discount rate (horizon) which is a value in the range } [0,1]$$

This paper introduced the idea of *local trail updates* and *global trail updates*. The purpose of the former was to try and diversify the pheromone matrix as using global trail updates alone was found to converge the matrix prematurely. The other rules were all very similar to those of Ant System. The most interesting contribution of the paper was the definition of the λ-branching factor.

The λ-branching factor (λ_{bf}) calculates an estimate of the size of the search space being focused on by the algorithm at any point in time. The equation for this is given in Equation 2.10. The number of values above λ_{bf} in each row of the pheromone matrix give the branching factor for that vertex, λ_v. Multiplying for all vertices in the construction graph one calculates the current search space. The parameter λ is specified by the researcher and acts as a threshold level, which is set to a value in the range $[0,1]$.

$$\lambda_{bf}(t) \longleftarrow \lambda\cdot(\tau_{max}(i,j) - \tau_{min}(i,j)) + \tau_{min}(i,j)$$
$$\tau_{max}, \tau_{min} \text{ are the extreme pheromone intensity values in the matrix } \tau \qquad (2.10)$$

Many later algorithms refer to this factor, stating that as λ_{bf} decreases the algorithm is focusing on a monotonically decreasing search space, which is presented as a beneficial quality of Ant algorithms. Various values for λ described in experiments are 0.04,0.06,0.08, and 0.1.

An extra parameter was included in the probability rule called q_0. This parameter decided whether the probability rule, at a particular point in the construction of a solution, used a uniform probability distribution, or was proportional to $\tau_{ij}\eta_{ij}$ as in Ant System. The inclusion of this parameter was an attempt to regulate exploitation versus exploration.

In [Gambardella and Dorigo, 1995] Ant-Q was compared to Ant System on a number of problems and was generally found to be better in terms of the mean solution found but the best results for each algorithm were the same. The problem with the results is

that only a few problems were given and little detailed analysis was performed on the data.

2.2.3 Ant Colony System

In 1996 the Ant System was given a relaunch as Ant Colony System (ACS) and applied to discrete optimisation problems [Gambardella and Dorigo, 1996] including QAP [Maniezzo and Colorni, 1999a], TSP [Dorigo and Gambardella, 1997; Gambardella and Dorigo, 1996] and Sequential Ordering Problems [Gambardella and Dorigo, 1997]. The algorithm merged a number of aspects of Ant-Q and Ant System together, and because of this overviews of the field tend to place the cited applications together under the name of ACS. A number of changes were made to the original Ant System, resulting from the discussion over the balance that needed to be made between exploration and exploitation and trying to prevent premature convergence.

The main two changes were: q_0, and the idea of two update procedures. The update rule from Ant System was split in two, one equation for local updates (2.11) and one for global updates (2.12).

$$\tau(i,j) \longleftarrow (1-\rho)\cdot\tau(i,j)+\rho\tau_0$$
where τ_0 is the parameter specifying the initial value of the pheromone matrix (2.11)

$$\tau(i,j) \longleftarrow (1-\rho)\cdot\tau(i,j)+\rho\frac{1}{L_{best}}$$
where L_{best} is the length of the best solution so far (2.12)

A variety of applications have used this version of the algorithm with and without local search. The following papers give an indication of the types of applications the algorithm was used for and whether a local search method was required to achieve satisfactory results.

In [Ellabib et al., 2002] ACS was applied to the Vehicle Routing Problem with Time Windows (VRPTW). No local search method was used but it was still competitive with some of the other metaheuristic techniques, which were also used without local search.

ACS has also been applied to Assembly Line Balancing in [Bautista and Pereira, 2002].

This did use a local search procedure to enhance solutions. The algorithm was successful with a variety of heuristics at competing with the current best.

An ACS algorithm was chosen in [Silva et al., 2002] to be applied to Logistic Process Optimising. In this investigation a simulator was used to investigate how the parameters influenced the quality of the schedules produced. These experiments were supported by real-world data. The assignment achieved by ACS proved to be better than the method used previously.

2.2.4 AS_{rank}

AS_{rank}, published in [Bullnheimer et al., 1997c], altered the update rules by introducing the concept of rank among a group of ant solutions. This was an idea that may have its roots in Genetic Algorithms where rank also played a part. The underlying idea was that the objective values were not always a clear indicator of strength. For instance a solution with an objective value of 19 compared with a solution with an objective value of 25 may not actually be that much better if there are no solutions with objective values in between. Therefore it made more sense to rate the solutions according to rank and eliminate this misleading sense of quality.

The notation used in the paper can be confusing as μ and σ would normally be associated with Statistical Analysis, however in this paper they are simply used to denote solutions with particular characteristics. With this in mind the notation is given below and is followed by the altered update rule in Equation 2.13.

- μ is the rank of an ant by fitness, for instance $\mu = 1$ points to the ant with rank of 1.
- σ_{best} is the fitness of the best ant found so far.
- ω is the number of ants to rank (set to $\sigma - 1$).
- $\Delta\tau_{ij}^{\mu}$ is the increase of trail intensity on edge (i, j) caused by the μ-th best ant.
- L_{μ} is the tour length of the μ-th best ant.
- $\Delta\tau_{ij}^{*}$ is the increase of the trail intensity on edge (i, j) caused by the σ elitist ants.
- σ is the number of elitist ants. Elitist ants are those that are allowed to imprint on the pheromone matrix.

- L^* is the tour length of best solution found.
- Q measures the influence of the new information relative to the influence of the initial trail level.

$$\begin{aligned}
&\tau_{ij}(t+1) \longleftarrow \rho\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau^*_{ij} \\
&\text{where } \Delta\tau_{ij} = \textstyle\sum_{\mu=1}^{\sigma-1} \Delta\tau^{\mu}_{ij} \\
&\text{and } \Delta\tau^{\mu}_{ij} = \begin{cases} (\sigma_{best} - \mu)\frac{Q}{L_\mu} & \text{if the } \mu\text{-th best ant travels on edge } (i,j) \\ 0 & \text{otherwise} \end{cases} \\
&\text{and } \Delta\tau^*_{ij} = \begin{cases} \sigma\frac{Q}{L^*} & \text{if edge } (i,j) \text{ is part of the best solution found} \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{2.13}$$

The algorithm was first introduced when applied to the TSP, followed by application to the Vehicle Routing Problem [Bullnheimer et al., 1997b,a] and then the Vehicle Routing Problem with Backhauls and Time Windows (VRPBTW) in [Reimann et al., 2002]. The introduction of rank did improve the performance when compared to that of Ant System on the TSP. Unfortunately the paper lacked information on specific experiments and so it is hard to compare the two approaches. All of the Ant algorithms were used with the parameters $\rho = 0.5$,$\alpha = 1$,$\beta = 5$ and $Q = 100$, highlighting the reliance on the heuristic for this algorithm. The relatively low value for ρ means that there was little reliance on previous solutions.

2.2.5 Fast Ant System

This algorithm is the Fast Ant System (FANT) for the Quadratic Assignment Problem [Taillard and Gambardella, 1997a]. It was a hybrid of a specialised Ant System with a local search. Its goal was to be as fast as possible, therefore any modifications have been made with this in mind.

The pheromone matrix was defined differently from that of the Ant System in that its values were not related to the objective value of the solutions, instead they were related to the difference of the constructed solution to the best one found so far. Let π^* be the best solution found so far and π be the current solution, and let $r^* = 4$ and r vary. At the beginning $r = 1$ and $\tau_{ij} = r, 1 \leq i, j \leq n$, meaning that the memory does not contain any information initially. The entries of the matrix τ are updated as in Procedure `FANT Memory Update Rule`.

Procedure FANT Memory Update Rule

/*** *Exploitation Mechanism* ***/;

if π^* *has improved* **then**

$r = 1$;

$\tau_{ij} = 1$, for all (i,j);

else if $\pi = \pi^*$ **then**

/*** *Diversification Mechanism* ***/;

$r = r + 1$;

$\tau_{ij} = r$ for all (i,j);

else

/*** *Usual Situation* ***/;

for $i = 1$ *to* n **do**

1a) $\tau_{i\pi_i} = \tau_{i\pi_i} + r$;

1b) $\tau_{i\pi_i^*} = \tau_{i\pi_i^*} + r^*$;

endfor

endif

The generation of solutions also differs slightly in that the construction process was random rather than sequential. This procedure can be found in Procedure Construction of a provisory solution.

Procedure Construction of a provisory solution

$I = \emptyset$; $J = \emptyset$;

while $|I| < n$ **do**

2a) Choose i, randomly, uniformly, $1 \leq i \leq n, i \notin I$;

2b) Choose j, randomly, $1 \leq j \leq n, j \notin J$, with probability $\frac{\tau_{ij}}{\sum_{1 \leq k \leq n, k \in j} \tau_{ik}}$ and set $\pi_i = j$;

2c) $I = I \cup \{i\}$, $J = J \cup \{j\}$;

endw

The solution is then passed through a non-deterministic fast descent algorithm twice to improve π;

As the whole point of this algorithm is to be fast, only one solution is created per cycle. In terms of performance FANT occasionally did better than other techniques such as HAS-QAP [Gambardella et al., 1999b], especially on larger problems.

2.2.6 MAX-MIN Ant System

The Max-Min Ant System (MMAS) [Stützle and Hoos, 1997, 1998a, 2000] was initially applied to the TSP and QAP [Stützle, 1997; Stützle and Hoos, 1998b] and enjoyed greater success than the original Ant System algorithm. The motivation for the Max-Min Ant System were results gathered using the original Ant System, where a number of important elements were noticed; these are listed below.

- More greediness improved performance and enabled the algorithm to find better solutions faster.
- The main problem of elitist methods was premature convergence.
- The best solutions were found when the λ branching factor was close to one, but when the λ branching factor was one then the search stagnated prematurely.

Therefore the algorithm differed from Ant System in three aspects: only one ant was allowed to update the pheromone matrix at the end of every iteration, the trail intensities were limited to some range τ_{min} and τ_{max}, and the trails were initialised in a certain way and were interpreted very specifically.

The modified update rule used by MMAS required only one ant to update the matrix every iteration. This ant imprinted either the *global best* (gb) or the *local best* (lb) solution. The rule is shown in Equation 2.14.

$$\tau_{ij}(t) \longleftarrow \rho \cdot \tau_{ij}(t-1) + \Delta\tau_{ij}^{best} \tag{2.14}$$

where τ_{ij}^{best} was $\frac{1}{L_{best}}$. Exactly which ant updated the matrix was dependent on the researcher, some always used the global best, while others only used the global best every *k* iterations.

The maximum limit of the trail intensity was set by a recursive formula, as shown in Equation 2.15, with the theoretical maximum being calculated by substituting L_{best} with the theoretical lower bound. τ_{min}, found in Equation 2.16, was calculated with consideration to a number of assumptions. Firstly it was assumed that the best tours would be found just before stagnation and that, more importantly, better tours were to be found near to the best tours. Through experimentation this property has been shown to be reasonable for the TSP benchmark problems. The other assumption was that the

main influence on the tour construction was the relationship between the upper and lower trail limits.

$$\tau_{max}(t) = \frac{1}{1-\rho} \cdot \frac{1}{L_{best}} \tag{2.15}$$

Convergence was said to have been achieved when the best tour found had been constructed with a probability significantly higher than zero; this probability p_{best} was set by the researcher. It was assumed at convergence that the best trail would have a trail intensity of τ_{max} and the rest would have an intensity of τ_{min}.

$$\tau_{min}(t) = \frac{\tau_{max} \cdot (1-\rho^{dec})}{avg \cdot p^{dec}}$$
where $p^{dec} = \sqrt[n-1]{p_{best}}$ and avg is the average number of options the ant has to choose at any decision point
(2.16)

Finally the trails were initialised by setting all of the trail intensities to the maximum pheromone value (τ_{max}). This led to the interpretation of ρ as the learning rate of the algorithm, as high values of ρ caused the pheromone values of undesirable arcs to fall more quickly, thus the algorithm converged more rapidly. The main parameters which could be used to adjust search tactics and the rate of convergence were identified as β, ρ and the number of ants, m.

MMAS has been applied to a number of applications, some of which are the same as Ant System. However, others have entered new territory and a number of these will be described here. In [Stützle and Dorigo, 1999b] MMAS was applied to the TSP and was compared to Iterated Local Search. *N*TS: CHECK THESE STATISTICS For nine problems between 198-1577 cities MMAS found a better solution than the other algorithms used for 77% of the problems on an average run. However, as a whole, MMAS only achieved a better quality solution in only 33% of the runs.

MMAS was used in [Stützle, 1998a] to attack the Flow Shop Problem. For this problem one of two local searches was used to optimise the solution depending on the size of the problem. The first was a swapping search that took $O(n^2m)$ to compute in a neighbourhood of $(n-1)^2$ and the second was a first-improvement strategy, which most of the time even out-performed the best-improvement version. For this the parameters were set such that few iterations would be performed, so ρ was set at 0.75

and p is set at $\frac{n-4}{n}$ and $\tau_{min} = \frac{\tau_{max}}{5}$. The algorithm outperformed a number of other methods such as Simulated Annealing and Multiple Descent.

MMAS was applied to the University Timetabling Problem in [Socha et al., 2002, 2003]. The paper showed that the algorithm performed better at a set of problem instances than an algorithm using the local search with random starting solutions. The algorithm used no heuristics when used with local search and relied solely on the pheromone matrix. A number of representations were used, but the best one was the event to room-specific time-slot, which was the most direct encoding of the problem.

In [Blum, 2002a], MMAS was applied to the Group Shop Scheduling Problem, in an effort to investigate intensification and diversification methods. The investigation compared MMAS to the use of elitist lists of solutions. In the paper [Socha, 2003] the effect of run-time on the choice of parameters was examined. The author discussed attempting to adapt the parameters to make them run-time independent without significant conclusion.

Two final applications of MMAS were to 2D HP Protein Folding Problem in [Shmygelska et al., 2002] and High-Level Synthesis in [Keinprasit and Chongstitvatana, 2004]. Both were novel applications where MMAS performed competitively especially as problems increased in size. When compared to Genetic Algorithms the researchers both benefited in a practical sense from not having to define specific crossover operators.

Finally the paper by [Stützle and Dorigo, 2002] showed a short convergence proof for Ant algorithms such as MMAS. The proof is not as strong as [Gutjahr, 2000] but sets out a number of smaller claims. First, that the algorithm will converge at infinity to the optimal solution. At this point the matrix will tend to τ_{min} except for those components contained in the optimal solution whose values converge to τ_{max}. However, it did not give any new ideas about run-time. An interesting corollary that came from this paper was that because only solutions better than the current best were allowed to update the matrix, the algorithm converged with probability 1. This prevents the algorithm swapping between two global optima, which would be allowed if solutions better than *or equal* to the global best were allowed to update the pheromone matrix. Essentially the argument for convergence was to prove that given enough time the algorithm would converge to the optimum because τ_{min} was never allowed to fall to 0.

2.2.7 Approximated Nondeterministic Tree Search

Approximated Nondeterministic Tree Search (ANTS) was proposed in [Maniezzo, 1998] and was applied to the Quadratic Assignment Problem. The algorithm proposed a number of modifications to the original Ant System. The key innovation was that at each step in the construction algorithm a lower bound could be given for the final solution. Using this lower bound (LB), which should be made as tight as possible, any solution whose lower bound was not below that of the current best solution was discarded. One of the benefits of this method was that the algorithm avoided stagnation by removing the parameter ρ, which was replaced by a moving average as shown in Equation 2.17.

$$\begin{aligned} &\tau_{ij}(t) = \tau_{ij}(t-1) + \Delta\tau_{ij} \\ &\text{where } \Delta\tau_{ij} = \tau_0 \cdot \left(1 - \tfrac{z_{curr} - LB}{\overline{z} - LB}\right) \\ &\text{where } \overline{z} \text{ is the average of the last } k \text{ solutions} \end{aligned} \tag{2.17}$$

In [Maniezzo et al., 2001] this algorithm was applied successfully to Data Warehouse Logical Design and in [Maniezzo et al., 2004] it was applied to Membership Overlay Design.

2.2.8 Ant Colony Optimisation

In the literature the use of Ant Colony Optimisation (ACO) and Ant System can be confusing. Ant Colony Optimisation is the metaheuristic and the implementations are either variants of Ant System or Ant Colony System. This subsection will be devoted to applications of the metaheuristic that do not have any specifically different algorithm characteristics. The reason this section is placed here is that chronologically, in terms of the algorithm variants, this is where the first papers started quoting the term Ant Colony Optimisation.

The first papers that started to quote ACO in their titles were from 1999. In [Michel and Middendorf, 1999] ACO was applied to the Shortest Common Super-sequence Problem. The algorithmic solution of string problems is important because many objects or processes in nature can be represented in an abstract way by strings of characters, one example of which is DNA encoding. The idea of this problem is to find a string that is a good representative for a set of strings. The solution to a problem is the shortest

common string that embodies the set; this is called the *super-sequence*. Like all the applications of ACO this is a NP-Hard problem. Therefore it is not surprising that Branch-and-Bound has also been applied to this problem; however, because the solution space of the problem grows very quickly Branch-and-Bound can only be used for short strings with small alphabets. The results using ACO were promising and this application would be of great value to help solve. It should also be mentioned that an island model was used to generate solutions.

Another early adopter of the term was [Varela and Sinclair, 1999] in which the author describes ACO applied to Virtual Wavelength Allocation. This used ACO to choose the combination of routing, fibre choice and wavelength allocation that comprise the optical-path-layer design for multi-wavelength all-optical transport networks. The quality of the solutions found was not discussed.

A more popular optimisation problem is Job Shop Scheduling which was the subject of [van der Zwaan and Marques, 1999]. The results for this investigation seemed promising, although the emphasis was on investigating suitable parameters for the Ant algorithm. However it did not produce any new results on either topic.

Another popular application for metaheuristics and probabilistic approaches is learning fuzzy rules. This was approached in [Cordon et al., 2000a] and involved learning a set of IF...THEN rules for a set of data for purposes of classification. The results were compared to those achieved previously with Simulated Annealing and Genetic Algorithms. ACO performed well, achieving results more quickly and sometimes of higher quality than these other approaches.

Two more scheduling related applications were investigated: the Single Machine Total Weighted Tardiness Problem (SMTWTP) in [Merkle and Middendorf, 2000; Merkle et al., 2000b] and Project Scheduling in [Merkle et al., 2000a]. Another group was also working on the SMTWTP [Stützle et al., 2000] at the same time. In the former study some new heuristics were trialled but both research groups obtained competitive results when the ACO algorithm was applied to these problems. The paper [Merkle and Middendorf, 2001, 2002a] returned again to the SMTWTP, but applied a new probability rule that is stated in Equation 2.18 called the *Relative Pheromone Evaluation Rule*. The objective of this new rule was to overcome bias between sampled and unused relations. For instance, for item j, if it has never been placed on a location i after a while it may never be picked, as some other item k will always have a higher pheromone value

for location i. Therefore the role of this equation was to normalise the pheromone values for a particular item by spreading good pheromone values into locations that have barely been used. The results of the experiments showed that, using their chosen parameters, the algorithm benefited from this new rule.

$$p_{ij} = \frac{\tau^*_{ij} \cdot \eta_{ij}}{\sum_{h \in S} \tau^*_{ih} \cdot \eta_{ih}} \quad \tau^*_{ij} = \left(\frac{\sum_{k=1}^{i} \tau_{kj}}{\sum_{k=1}^{n} \tau_{kj}} \right)^{\gamma} \cdot \tau_{ij} \tag{2.18}$$

Both [Guntsch and Middendorf, 2001; Guntsch et al., 2001] applied ACO to the problem of the Insert/Delete Dynamic TSP. The most interesting aspect of these papers was the methods that were used to reset the pheromone matrix when a change was detected. In total three methods were proposed. The first was the *Restart strategy*, which modified the pheromone matrix by a fixed amount using Equation 2.19 but with a γ_i equal to a fixed value for all cities. However, this is limited as it does not take into account where the change in the problem occurred. The other two strategies specialised the decay factor to each city (γ_i). The update rule, shown in Equation 2.19, was then applied to the pheromone matrix.

$$\tau_{ij} = (1 - \gamma_i)\tau_{ij} + \gamma_i \frac{1}{n-1} \tag{2.19}$$

The second strategy was called the η-*Strategy*. In this strategy the city i was assigned a value γ_i proportionate to its distance from the inserted/deleted city j. This distance d^{η}_{ij}, in Equation 2.20, was derived from the heuristic value, η_{ij}, in such a way that a high η_{ij} implied a high d^{η}_{ij} and that scaling the η-values had no effect.

$$d^{\eta}_{ij} = 1 - \frac{\eta_{avg}}{\lambda_E \cdot \eta_{ij}} \tag{2.20}$$

where $\eta_{avg} = \frac{1}{n-1} \sum_{i=1}^{n} \sum_{k \neq i} \eta_{ik}$ and the strategy-specific parameter $\lambda_E \in [0, \infty)$ which scaled the width of the affected area. A city i was then assigned the value $\gamma_i = \max(0, d^{\eta}_{ij})$.

The third strategy was the τ-*Strategy*. This strategy used a distance measure based on the pheromone intensity to calculate the coefficient γ_i. The pheromone intensity between two cities i and k was d^{τ}_{ik}, shown in Equation 2.21, and was defined as the maximum over all paths P_{ik} of the product of pheromone-values on the edges in P_{ik}.

$$d_{ik}^{\tau} = \max_{P_{ik}} \prod_{(x,y)\in P_{ik}} \frac{\tau_{xy}}{\tau_{max}} \tag{2.21}$$

In the case of an insertion of a city into the problem the pheromone values of the edges to the two closest cities were set to τ_{max}. A parameter $\lambda_\tau \in [0,\infty)$ was included to modify the scale if required. Therefore the value assigned to each city i was $\gamma_i = \min(1, \lambda_\tau \cdot d_{ij}^{\tau})$.

Another interesting modification was made to solve the problem of how to keep the best solution after cities have been modified. Instead of throwing away this solution it was modified by first removing any deleted cities and then connecting the predecessors to their successors. Next, this procedure added in any inserted cities where they minimised the increase in path length. This procedure was named *KeepElitist*.

In this investigation the Restart strategy and η-Strategy were found to be the best, followed by the τ-Strategy. In the latter paper by [Guntsch and Middendorf, 2001] a combination of these strategies was suggested. The state of the pheromone matrix was recorded using normalised entropy E, the formula for which is shown in Equation 2.22. E is a measure of the amount of information in the matrix. For instance, a matrix in which all values were the same would have a normalised entropy of 0, and in a matrix where each value was different it would be 1.

$$E = \frac{1}{n \log n} \sum_{i=1}^{n} \sum_{j=1}^{n} -\tau_{ij} \log(\tau_{ij}) \tag{2.22}$$

In [Roli et al., 2001] another application called, Maximal Constraint Problems, was proposed. In this paper the CSP graph (X,D,C) (where X was the set of variables, D was the set of values for each variable and C was the set of constraints) was translated into a graph with each vertex being a variable-value pair. Another attempt at a CSP solver with binary constraints was described in [van Hemert and Solnon, 2004] and this takes a similar approach. Both papers showed good results for small CSP problems, however the complete solvers were competitive, and it is with scale that the advantages of Ant algorithms were significant. Another attempt was undertaken in [Meyer and Ernst, 2004] where ACO was hybridised with a Constraint Propagation solver. This combination used the Constraint Propagation solver to do most of the work and when the search space was fairly large ACO would take over to try and find a good solution. As with the previous two attempts promising results were achieved.

Both [Bianchi et al., 2002a,b] dealt with an ACO application to the Probabilistic TSP. This problem is a variant of the TSP but each city has a probability of being visited. The objective is to find the best a priori tour whose expected tour length is minimised. In other words a tour in which if a city is removed, the tour is still likely to be minimal for all tours without that city. In this algorithm the basic Ant System was used but the main focus was to develop a powerful heuristic to drive the algorithm. They showed that combining the Ant System and their heuristic delivered a useful solver for this problem.

Another application to the TSP was given in [Montgomery and Randall, 2002] but the focus here was an investigation of anti-pheromone, or negative reinforcement. Three methods were proposed: Subtractive Anti-Pheromone (SAP), Preferential Anti-Pheromone (PAP) and Explorer Ants. Using Subtractive Anti-Pheromone pheromone was removed from those edges involved in the worst solution of a cycle. In Preferential Anti-Pheromone two pheromone matrices were used, one for positive reinforcement and the other for negative. These were combined in a weighted sum in the probability rule. Finally Explorer Ants were those ants that, instead of preferring pheromone values that were high, preferred low pheromone values. Approximately a fifth of the ants per cycle was found to be the best ratio of Explorer Ants. Unfortunately, these methods were only successful on small problems ($n < 100$) and none of them worked particularly well.

In [Dorigo et al., 2002b], to substantiate the theory behind Ant algorithms the update rules were derived from both Cross-Entropy and Stochastic Gradient Ascent algorithm theory. While the paper achieved this objective the practical value was limited as both of these methods have similar issues being debated as for Ant algorithms.

Both Static and Dynamic Multi-processor Scheduling were investigated in [Ritchie, 2003]. From speaking to the author it appears that the Ant algorithm acted as a procedure for disrupting the solution on which the Tabu Searches were to be used. Another application undertaken at the University of Edinburgh was the application of ACO to Bin Packing and Cutting Stock Problems in [Ducatelle and Levine, 2004]. The main omission to this attempt was that it had yet to combine the algorithm with a local search, thus it could not compete with the Genetic Algorithm implementations. However, it did show that ACO could be used with these problems. In addition, some interesting points about parameters emerged from these experiments. For instance the number of ants m was optimal when equal to the number of items in the problem. β

was identified as a crucial parameter the value of which could mean success and failure. However, the choice of good values for β was small ($\beta \in \{2,5,10\}$). In [Levine and Ducatelle, 2004], a local search was added and better performance was achieved.

Dynamic Vehicle Routing Problems are becoming more important due to the advance in communication and information technologies that allow information to be obtained and processed in real time. For this reason an application of ACO for this problem was suggested in [Montemanni et al., 2003]. The algorithm was part of a larger system that takes a snapshot of the problem. The algorithm then solves that snapshot until interrupted by a new event. Benchmarks were created for this problem and it was found that for this method to work well the emphasis had to be put on how easily good solutions could be transferred to a new snapshot.

In the same year another type of static Vehicle Routing Problem was demonstrated with ACO in [Donati et al., 2003]. This version was Time Dependent Vehicle Routing Problems (TDVRP). This problem has the added constraint that the time cost of a particular route changes depending on the time of day the vehicle sets out. This paper suggested a model better suited to this style of problem and investigated the impact of this new constraint on solutions initially created using the ordinary Vehicle Routing Problem.

[Gutjahr, 2003a, 2004] proposed a converging algorithm using ACO and Monte Carlo sampling. This was used to estimate the objective value of a solution to a Stochastic Combinatorial Optimisation problem, in this particular case TSP with Stochastic Time Windows. The main change to the algorithm was that the objective function required a sample taken to get a mean fitness for a particular solution. This implementation was called S-ACO and was based on Ant System. It performed well on selected problems when compared to a vanilla Stochastic Simulated Annealing approach.

The Set Packing Problem was the focus of [Gandibleux et al., 2004]. This paper split ACO into three procedures: one for exploitation, one for exploration and one greedy routine. Initially the greedy routine was run to establish a lower bound. From then on the choice was between the exploration and exploitation procedures. These were selected probabilistically, so that as more cycles were performed the probability of exploitation increased. When compared to GRASP the ACO algorithm achieved better results.

[Acan, 2004] investigated the ability of an external memory to store parts of solutions

which were then used in the construction process. The mechanism was applied to TSP and QAP with results showing an improvement compared to MMAS. This was an interesting approach as a great deal of time of an algorithm based on the ACO framework is spent in the construction of new solutions. This work demonstrated that this was a viable way to help scale up algorithms that use a construction procedure. The approach is given in Figure 2.14.

Figure 2.14: Algorithm for using an External Memory [Acan, 2004]

```
1  Initialise the external memory;
2  repeat
3      Construct m solutions;
4      Rank the solutions by objective function values;
5      foreach s in top k-best do
6          Cut a randomly-positioned and randomly-sized segment and insert it
7          into memory;
8      endfch
9      Update the pheromone matrix
10 until memory is full;
11 while stopping criteria not reached do
12     Let all ants select a segment from memory using tournament selection;
13     Let all ants complete a valid solution using the segment retrieved
14     from memory;
15     Update Pheromone Matrix;
16     foreach s in top k-best do
17         Cut a randomly-positioned and randomly-sized segment and insert it
18         into memory removing worst segment if necessary;
19     endfch
20 endw
```

Competition Controlled Pheromone Update was proposed in [Merkle and Middendorf, 2004]. This method allowed good solutions that were found during periods of greater competition to be given more pheromone. A measure of competition based on Kullback-Leibler distances was introduced, shown in Equation 2.23. $\sigma_{ij}^{(m)}$ was the probability that the best of m ants in a cycle selects item j for place i. Once more this addition did appear to improve the solution quality compared to the ordinary update

rule, but as only a few instances were selected a definitive judgement was not possible.

$$d_i = \sum_j \sigma_{ij}^{(1)} \cdot \log_2 \left(\frac{\sigma_{ij}^{(1)}}{\sigma_{ij}^{(m)}} \right) \quad \text{where } i \in [1:n] \tag{2.23}$$

Deception and *bias* are two concepts found in Evolutionary Computation. Deception occurs when an algorithm is misled due to features of the particular problem instance, whereas bias is due to the solution representation. In [Blum and Dorigo, 2004] these two concepts were applied to ACO. Deception was relabelled *first order deception* while bias was renamed *second order deception*.

A First Order Deception System (FODS) occurs if there exists an initial setting of pheromone values such that the algorithm does not, in expectation, converge to a globally optimal solution. This expectation was calculated according to the model proposed in [Merkle and Middendorf, 2002b]. The model was a deterministic version of ACO formed by using the expected update value for a particular iteration. The practical consequence of this was to show that ACO was susceptible to the same sorts of deception as Evolutionary Computation (EC) algorithms. An example of this would be any problem with two or more stable fix-points (a state of a system where once in that state the algorithm can never leave) with at least one leading to only a local optima, for instance a n-bit trap function.

The other type of error it is possible for ACO to make is called Second Order Deception. This is when the expected pheromone values at time $t+1$ are less than the expected values at time t. This may occur in small time windows of varying size during the algorithm run and is particularly visible when updating the matrix using the iteration best solution. The end result can be that the average solution quality later in the algorithm is less than the average solution in the first iteration.

Related to this idea of misleading ACO, the paper [Montgomery et al., 2004] explored search bias in constructive metaheuristics with emphasis on its implications for the algorithm. Two kinds of bias were identified, *representation bias* and *construction bias*. The first is related to the fact that in some representations of a problem a particular solution will be represented in multiple ways, therefore ACO will be pushed towards this solution. Construction bias occurs when two nodes in the search tree are at the same height but have different amounts of branching. The effect of this is that solutions will be found more easily on the side of the tree with less branching. These two forms

of bias can interact in the search process, for example if an over-represented solution is in a very bushy part of the search tree then its probability of being found will decrease. The use of heuristics and local search methods in ACO can help to counteract these search biases.

The above paper also discussed the role of assignment order in problems such as QAP. The conclusion given was that the order alters the distribution of solutions in the search tree but does not created any additional biases. This means that the neighbours will change but the branching factor will not. It also discussed the role of assignment order in problems where large parts of the search space produced infeasible solutions and proceeded to state that heuristically chosen orderings could reduce the probability of constructing such a solution.

Beam-ACO was a hybrid introduced recently in [Blum, 2005], although it had already been attempted in [Maniezzo, 1998] with respect to the Quadratic Assignment Problem. It hybridises ACO with Beam Search (Figure 2.15), which is a method for restricting the branching factor when searching a tree structure. It was demonstrated on the Open-Shop Scheduling problem and proved to be very promising when compared to the standard ACO and Genetic Algorithm solvers. Beam Search acts as a generalised local search, as a neighbourhood ($N(\cdot)$) is defined and then deterministically searched with reference to the lower-bound to guide the search. The key to a successful application of Beam Search is a good lower bound, without which the candidate solution set would be too large. It was used with ACO to construct the solutions for the cycle, and took the form of the basic Beam Search algorithm with the $argmax\{\eta(c)|c \in N(s^p)\}$ on line 8 of the algorithm in Figure 2.15, replaced with the normal Ant System probability rule.

2.2.9 Hypercube Framework for ACO

In [Blum et al., 2001] a new way of looking at ACO was introduced called the Hypercube Framework. The framework equated ACO to Binary Integer Programming (BIP) by treating solutions as binary vectors of length n. In BIP one is given a set of decision variables that correspond to solution components, which can take the value 0, exclusion, or 1 (inferring inclusion of the component in the solution). One can represent this style of problem as a n-dimensional hypercube where each corner is a solution, as shown in Figure 2.16. BIP then relaxes the constraints of the problem to allow each

Figure 2.15: Beam Search Algorithm [Blum, 2005]

input : an empty partial solution $s^p = \langle\rangle$, beam width k_{bw}, maximum number of extensions k_{ext}

output: a set of candidate solution B_c

1 $B \leftarrow s^p, B_c \leftarrow \emptyset$;
2 **while** $B \neq \emptyset$ **do**
3 $B_{ext} \leftarrow \emptyset$;
4 **foreach** $s^p \in B$ **do**
5 count $\leftarrow 1$;
6 $N(s^p) \leftarrow$PreSelect$(N(s^p))$ {optional, PreSelect is a filtering function};
7 **while** *count* $\leq k_{ext}$ *AND* $N(s^p) \neq \emptyset$ **do**
8 Choose $c \leftarrow$argmax$\{\eta(c)|c \in N(s^p)\}$;
9 $s^{p'} \leftarrow$ extend s^p by adding component c;
10 $N(s^p) \leftarrow N(s^p) \setminus \{c\}$;
11 **if** $s^{p'}$ *extensible* **then**
12 $B_{ext} \leftarrow B_{ext} \cup \{s^{p'}\}$;
13 **else**
14 $B_c \leftarrow B_c \cup \{s^{p'}\}$;
15 **endif**
16 count $\leftarrow$ count $+ 1$;
17 **endw**
18 **endfch**
19 Rank the partial solutions in B_{ext} using a lower bound $LB(\cdot)$;
20 $B \leftarrow$ select the min$\{k_{bw}, |B_{ext}|\}$ highest ranked partial solutions from B_{ext};
21 **endw**

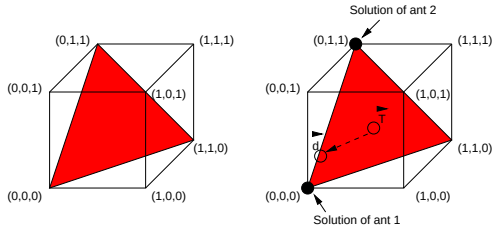


Figure 2.16: (Left) The set S of feasible solutions consists of the three vectors $(0,0,0)$, $(1,1,0)$ and $(0,1,1)$. The shaded area is the set $\vec{S}$. (Right) Two solutions are shown by black dots, created by two ants. $\vec{d}$ is the weighted average of these two solutions ((0,0,0) is of the higher quality) and $\vec{\tau}$ will be shifted towards $\vec{d}$. [Blum et al., 2001]

variable to take a real-valued number, thereby making a solution, from an extended set $\tilde{S}$ of solutions, as a convex combination of binary vectors. In Equation 2.24, the weights α_i are translated in an Ant System to be the pheromone value τ_i associated to component $o_i, 1 \leq i \leq n$.

$$\vec{v} \in \tilde{S} \Leftrightarrow \vec{v} = \sum_{\vec{v} \in S} \alpha_i \vec{v}_i, \quad \alpha_i \in [0,1] \tag{2.24}$$

The significance of this paper was that it defined two concepts: *global desirability* and *global frequency*. For every arc the framework specified two numbers: pheromone intensity, representing the desirability of an arc; and the number of times an arc has been traversed. This framework might be able to offer some method for pruning the search space; an ability that many incomplete algorithms of this sort currently lack.

This framework was used in several applications including the Edge-Weighted k-Cardinality problem [Blum, 2002b] and the First Order Shop Scheduling problem [Blum and Sampels, 2002a,b].

2.2.10 Graph-based Ant System

In [Gutjahr, 2000, 2002] a number of algorithms were proposed that had an associated theoretical proof of convergence, the main algorithm being known as Graph-based Ant System (GBAS). The resulting algorithm was a theoretical formalisation of the Ant System algorithm and showed that the algorithm converges to the optimal solution with a probability that can be made arbitrarily close to 1.

The formalisation follows that of the original Ant System with some modifications to

the update rule and the pheromone matrix. The pheromone matrix is initially set to $\frac{1}{\text{number of arcs}}$ for each arc (i, j) and $\sum_{\forall \text{edges } (i,j)} \tau_{ij} = 1$. The update equation is given in Equation 2.25.

$$\tau_{ij}(m+1) = (1-\rho)\tau_{ij}(m) + \rho \frac{1}{\sum_{\forall \text{edges}} \sum_{s=1}^{S} \Delta\tau_{kl}^{(s)}} \sum_{s=1}^{S} \Delta\tau_{ij}(s)$$
$$\text{where } \Delta_{ij}(s) = \begin{cases} \phi(f_s), & \text{if agent } A_s \text{ has traversed edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$$
m is the cycle and
$\phi(\cdot)$ is a function that maps objective values to an amount of positive reinforcement
(2.25)

To converge a set of the best solutions found so far are held and these are the only ones that update the pheromone matrix. This is similar to the global best update in MMAS.

There are a number of assumptions and technical changes in the algorithm, these will be explained in Chapter 5 when an implementation of GBAS is given.

2.2.11 Best-Worst Ant System

This algorithm, found in [Cordon et al., 2000b], again rooted in the Ant System, made three changes to try and improve diversity in the algorithm. The inspiration for these changes came from PBIL (Population Based Incremental Learning), another search algorithm described in Section 2.4.7.

The first change used both the best solution and the worst solution in the current cycle, the former to positively reinforce and the latter to negatively reinforce. In each cycle local search was applied to both solutions. Then the global best solution was used to update the pheromone matrix, similarly to ACS. The edges in the worst solution of the cycle that were not present in the global best solution were penalised by another application of the decay factor to the pheromone trail associated with those edges.

The next alteration was to include a restart of the search process when the algorithm had stagnated. Stagnation was defined to be the state at which a few values in the pheromone matrix were high while the remainder were almost at zero. The restart was performed by resetting the values to their original state when the percentage of edges that were in the current best and worst solutions of the cycle fell lower than a specific percentage, which was set as a parameter.

The final difference was that the pheromone matrix suffered mutations to introduce diversity into the search process. The mutation operator introduced small changes into early solutions and larger changes into later solutions, thus trying to kick the algorithm into unexplored territory. To accomplish the mutation Equation 2.26 was applied.

$$\tau'_{rs} = \begin{cases} \tau_{rs} + mut(it, \tau_{threshold}), & \text{if } a = 0 \\ \tau_{rs} - mut(it, \tau_{threshold}), & \text{if } a = 1 \end{cases}$$
$$\text{where} \quad \tau_{threshold} = \frac{\sum_{(r,s)\in S_{gb}} \tau_{rs}}{|S_{gb}|}$$
$$a \text{ is a random variable between 0 and 1}$$
$$it \text{ is the current iteration} \tag{2.26}$$
$$it_r \text{ is the last iteration where a restart was performed}$$
$$Nit \text{ is the maximum number of iterations of the algorithm}$$
$$mut(it, \tau_{threshold}) = \frac{it - it_r}{Nit - it_r} \cdot \sigma \cdot \tau_{threshold}$$

The mutation operator was reset each time a restart was performed, as the algorithm was starting with a new search. The parameter σ specified the power of the mutation with respect to the number of iterations into a particular restart.

The algorithm showed a strong exploitation characteristic as well as the potential for diverse exploration, making a good trade-off between these two concepts and achieving competitive search times. This first paper applied the algorithm to the TSP with significantly better results than both Ant System and ACS. In [Cordón et al., 2002] the algorithm was applied to QAP, again with successful results, and the authors found a hierarchy of importance between these three modifications (going from most to least important): restart, mutation and worst update.

2.2.12 Ant Programming and Model-based Search

[Birattari et al., 2002a] defined *Ant Programming* as the class of algorithms that dealt with optimisation problems using optimal control. The actual contribution was a reworking of the Ant metaphor in the style of a Markov representation, exploring the relationship between the exploded state-space graph and the limited graph representation of ACO.

A paper examining similar themes was [Blum et al., 2002], which stated that although positive feedback provided a mechanism for solution improvement, the rate of im-

provement decreased with time in certain settings. It used an update rule that iterated through all solutions generated in a cycle and updated them proportionally to their objective value. In the conclusion the authors stated that this decrease in improvement rate would not occur with a different update rule. Therefore the practical information that this paper offered was limited. It would have been more useful to explore how the rate of improvement changes given a more general feedback model.

[Zlochin and Dorigo, 2002] compared a number of model-based search algorithms such as Estimation of Distribution Algorithms, Stochastic Gradient Ascent, Cross-Entropy and ACO. This investigation confirmed the relative performance of these algorithms and the similarities in their origins and applications.

2.2.13 Population-based Algorithms

One of the natural modifications of Ant System was to introduce an explicit population that involved some level of *elitism*. Two such algorithms have been proposed: Population-based ACO (P-ACO) and Omicron ACO (O-ACO). Both have had reasonable success although the number of problems they have been applied to is limited.

2.2.13.1 Population-based ACO

The Population-based ACO algorithm was first applied to static TSP and QAP [Guntsch and Middendorf, 2002b] and then later to Dynamic versions of TSP and QAP [Guntsch and Middendorf, 2002a]. For this algorithm the first k solutions generated were placed into a population structure. No pheromone evaporation was performed and on the creation of the $k+1$-th solution another solution was removed from the population, its fitness subtracted from the pheromone matrix. The new solution was then placed in the population.

In the first paper the oldest solution was the one removed. The algorithm set a value τ_{max} that determined the maximum amount of pheromone to be placed in the matrix and was set using the following formula $\frac{(1-\omega_e)(\tau_{max}-\tau_{init})}{k}$, τ_{init} was set to $\frac{1}{n-1}$ (n being the number of cities in a TSP), or $\frac{1}{n}$ for QAP (n being the number of facilities). ω_e was the relative amount of pheromone added by the best ant found so far compared with that added by the best ant in the iteration. The algorithm performed as well as MMAS in the trials conducted in this paper.

Another implementation of a Population-Based ACO was given in [Scheuermann et al., 2004], where the algorithm is implemented in hardware on Field Programmable Gate Arrays. The paper stated that the application was successful but no further results were given.

2.2.13.2 Omicron ACO

In [Gómez and Barán, 2004] an algorithm called Omicron ACO (O-ACO) was described. The algorithm differed from MMAS in the way that the pheromone matrix was updated. The motivation for this was for it to be a simpler algorithm to study. In O-ACO a constant pheromone matrix τ^0 is defined with $\tau^0_{ij} = 1$, for all i and j. O-ACO then maintained a population $P = P_x$ of m individuals which corresponded to the best solutions found so far.

The initial population was chosen using τ^0 and then at every iteration a new individual P_{new} was generated, replacing the worst individual in P if it was better and also different from any other member in P. After K iterations, τ was recalculated by first setting $\tau = \tau^0$; then $\frac{o}{m}$ was added to each element in τ every time the arc (i, j) appeared in P, where o was some constant.

This algorithm was similar to the Population-based ACO described in Section 2.2.13.1, but it differed in two respects. Firstly, O-ACO was updated only every K iterations, whereas in P-ACO it is updated on every iteration ($K = 1$). Secondly, the population in O-ACO could not have duplicate solutions in its population.

Any conclusions drawn about ACO in this paper were domain dependent due to the fact that it concentrated primarily on the global convexity of the Symmetric Travelling Salesman Problem. The authors concluded the reason that elitist ACO works was that the search area was narrowed with the introduction of each new good solution and that good solutions lay near to each other.

2.2.14 Simple ACO

A simple framework was proposed in [Stützle and Dorigo, 2001] for investigating some of the basic properties of ACO. Here a number of the features of the original system were either reduced or removed. It contained no heuristic, making the probability rule

simpler. The update was also simplified to make the *differential path length* effect the prominent reason for improvements in solution quality. This effect was mentioned in Section 2.1.

The authors drew four conclusions. Firstly, that the differential path length effect was not enough to solve large optimisation problems. Secondly, that solution quality based pheromone update was important to allow a fast convergence of S-ACO. The third finding was that large α values led to disproportionate emphasis on initial random fluctuations which led to bad algorithm behaviour. Finally, pheromone evaporation was important when trying to solve more complex problems.

These conclusions are interesting to some degree. The first is a realisation of the paradox that a body of evidence is required for a good solution to get the necessary reinforcement but reinforcement is required to acquire the body of evidence. The second is not necessarily supported due to the fact that updating pheromone in relation to solution quality is fine if the domain is continuous, but given a discrete domain it is unwise and leads to the bias mentioned earlier in [Blum and Dorigo, 2004]. The third conclusion is the most useful in the sense that it warns against relying on evidence too early in the search process. This is important as the boot strap nature of the algorithm means that these early mistakes can play a large part in causing early convergence. The final conclusion is drawn with the only comparison being no pheromone evaporation and not whether ρ was a good way to perform this operation, which limited the significance of this conclusion.

2.2.15 Continuous Combinatorial Optimisation

The Ant metaphor has not been restricted to discrete optimisation problems but has also been applied to continuous problems, with some modification. [Socha, 2004] introduced a straightforward transference of the ordinary discrete algorithm to a continuous model, moving to a mixture of Gaussians to compute probabilities. Updates were achieved by adding a Gaussian which was centred on the best solution of that iteration or on the global best. Evaporation was done in a number of ways: by subtracting a fixed Gaussian, via modifying the standard deviation, and by altering the weights in the mixture model.

Another variant called Continuous Ant Colony System (CACS) was proposed in [Pour-

takdoust and Nobahari, 2004]. It modelled the pheromone matrix as a set of Gaussians with the update requiring a modification of the standard deviation. The results were comparable to those of a Genetic Algorithm or API approach (see Subsection 2.4.1).

2.2.16 Parallel and Multiple Colony Ant Implementations

One of the many adaptations of Ant algorithms has been to use multiple ant colonies, or to run the ants, in parallel. Depending on the implementation it is clear that the most important question is how to perform communication between the ants and/or colonies.

The first paper to look at this question used a simple parallelisation of the Ant System applied to the TSP [Bullnheimer et al., 1997d]. There were two proposed implementation strategies, synchronous and partially asynchronous. The former was the straightforward practice of computing the TSP tours in parallel resulting in a asymptotic speed-up, assuming an infinite number of processing elements and one processing element per worker. This ignored communication overhead and resulted in high frequency and volume of data. This was because, after each iteration, all the tour information and their lengths had to be transmitted to a master process that then computed alterations to the pheromone matrix.

In the partially asynchronous version each worker was a separate colony and results were merged after a set number of iterations. This meant more work being done before the master process computed the global information. These results raised some important issues and showed that the parallelisation of Ant algorithms was possible but not necessarily as trivial as had been predicted.

A similar route was taken in [Michel and Middendorf, 1998] where an algorithm was introduced that finds parameters for a heuristic for the Shortest Super-sequence Problem. In this algorithm an island model was used and the islands share trail information every few iterations.

One obvious alteration would be to perform the local search in parallel, as it is usually very resource intensive. MMAS was parallelised in this way in [Stützle, 1998b], also making use of a master-slave approach, and applied to the TSP. The author concluded that this can be a highly effective strategy for increasing the rate of optimisation.

Multiple Ant Colonies were also used in [Gambardella et al., 1999a] applied to the

Vehicle Routing Problem with Time Windows (VRPTW). Each colony optimised a certain objective function and the solutions were then combined in a master colony. Using this method the algorithm has been shown to be comparable to the best known methods in speed and solution quality.

In [Middendorf et al., 2000] a number of different communication mechanisms were investigated using the TSP problem. Each colony was on a separate processor and the colonies were formed into a ring of resources. The following mechanisms were experimented with:

- Exchange of global bests between all colonies.
- Circular exchange of local best solutions.
- Circular exchange of migrants (same as the previous but only using best half of solutions).
- A combination of the previous two options.

The best solution found was to exchange infrequently the locally best solution found with the neighbouring colony. This optimised the trade-off between communication and solution quality.

2.2.17 Ants applied to Networks

One of the most successful applications of these algorithms has been to data packet routing. Two groups were working on this application in 1998. The first, AntNet [Di Caro and Dorigo, 1997, 1998d,c,b,a,e], produced very good results compared with the more traditional routing methods. The second [Bonabeau et al., 1998] was an extension of work done by [Appleby and Steward, 1994; Schoonderwoerd et al., 1996] that combined "smart" ant-like agents with dynamic programming.

Another networking application was proposed in [Montresor, 2001]. This method used ant-style searching to locate resources for use with a peer-to-peer system.

AntNet was the basis for another set of routing algorithms for sensor networks set out in [Zhang et al., 2004]. This has a number of important differences compared to communication networks. Firstly, the address-based specification of destinations was replaced with a feature-based specification, including geographical location and information gain. Secondly, the routing metric used was not just shortest delay, but also

included energy usage and information density. Finally there were more major traffic patterns in sensor networks, which include peer-to-peer, multi-cast (one-to-many) and converge-cast (many-to-one).

In [Nowé et al., 2004] a Multi-type Ant Colony System was proposed. This used different types of ants to develop similar but competing solutions to a problem. The algorithm was applied to the problem of finding disjoint paths in graphs. The idea was that ants of a similar type would be attracted to the pheromone of one another and those of a differing type would be repulsed, thereby modelling competition and co-operation explicitly. Some possible applications of this have been finding disjoint routes in a network, production planning and traffic engineering. The success of this model has yet to be published.

2.3 Parameter Selection

Parameter selection has played an important role throughout the application of Ant algorithms. As has been described the motivation for many of the emerging algorithms has been to limit, or understand, the number of parameters the algorithm requires to get the best performance. In this section the parameter ranges are studied, some ways of automatically determining these are looked at and finally the conclusions drawn from these studies will be formalised.

In all the algorithms there are common decisions that must be made. These decisions are realised with the introduction of certain parameters. A general list of parameters and their notation can be found in Table 2.4.

An attempt at evolving the parameters for Ant Colony Optimisation was given in [Botee and Bonabeau, 1998]. Here the authors created a Genetic Algorithm to evolve the parameters for a version of Ant Colony System applied to the TSP. There had been a previous attempt by [White et al., 1998] but this was applied to routing and therefore could not be easily compared using common benchmarks. The parameters and their ranges have been given in Table 2.5 and from this the scale of the problem is immediately apparent.

The fitness function used in the Genetic Algorithm was a weighted sum of four components: $F = \sum_{i=1}^{4} c_i \cdot F_i$, where c_i was the weight for the component i and F_i refers to one of the functions in the list below.

Parameter	Description
α	The weighting given to the pheromone trail intensities.
β	The weighting given to the visibility, or heuristic knowledge.
ρ	The weighting given to $\Delta\tau$.
$(1-\rho)$	The weighting given to τ_{old}.
γ	The ratio of using the iteration best and global best to update the pheromone matrix.
γ_n	The number of solutions to update pheromone matrix with per cycle.
$\overline{\tau}_{min}$	The minimum values that the pheromone intensity can reach.
$\overline{\tau}_{max}$	The maximum values that the pheromone intensity can reach.
$\overline{\tau}_0$	The initial values of the pheromone matrix.
m	The number of ants used per cycle.
max_{cycle}	The maximum number of cycles the algorithm is run for.

Table 2.4: Canonical Set of Parameters for Ant Algorithms

Parameter	Minimum	Maximum	Description
m	1	$2n$	Number of ants
q_0	0.0	1.0	Exploitation Probability
α	0.0	5.0	Pheromone Trail weighting
β	0.0	10.0	Heuristic weighting
ρ_{local}	0.0	1.0	Local trail decay
ρ_{global}	0.0	1.0	Global trail decay
Q	0.0	100.0	Scales global trail to local trail update ratio
γ	0.1	3.0	Amplifies shorter tours on update
τ_0	0.0	0.5	Initial pheromone intensities
a	0	9	Local search parameter
b	0.0	1.0	Local search parameter
n_l	1	1	Number of nearest neighbours (not used)

Table 2.5: Evolved Parameters in [Botee and Bonabeau, 1998]

Parameter	Values
α	$\{1, 1.25, 1.5, 2\}$
m	$\{1, 5, 10, 25\}$
β	$\{0, 1, 3, 5\}$
ρ	$\{0.6, 0.7, 0.8, 0.9\}$

Table 2.6: Evolved Parameters in [Birattari et al., 2002b]

- $F_1 = \frac{1}{L+1-L_+}$, where L was the best tour found by the colony and L_+ was the best tour length found by all of the colonies thus far, c_1 was set in the range $[2.0, 3.0]$.
- $F_2 = \exp^{-\frac{v}{5 \cdot n}}$, where v was the iteration in which the best tour was found, reflecting the fact that it was good that a tour was found quickly; c_2 was set in the range $[0.5 - 0.8]$.
- $F_3 = \exp^{-\frac{m}{10 \cdot n}}$ encouraged m to be as small as possible, to maximise efficiency; c_3 was set to the value $\frac{0.5}{m}$.
- $F_4 = \exp^{-\frac{\sigma}{m}}$ was to act on the local search and tried to minimise the amount of work done by the 2-Opt local search method; c_4 was set in the range $\frac{[0.2, 0.5]}{\sigma}$.

The method did produce good results and challenged the human set parameters of previous attempts. However, the main downside was the computational effort involved; having to run ACS on each problem forty times per generation and the GA was given 100 iterations with a population of 40.

The parameter configuration method proposed in [Birattari et al., 2002b] was used on MMAS applied to the TSP. The idea was to perform model selection using cross-validation to try and gather evidence for and against particular scenarios until a winning parameter scenario was left (the parameters chosen are shown in Table 2.6). The results were vague but encouraging. The algorithm seemed to produce better parameter sets than those chosen by hand, although the work put into finding these sets was unreported.

The work proposed in [Randall, 2004] was interesting as it described a Meta Ant System that tried to optimise parameters for the underlying ACS-TSP algorithm, while at the same time solving the particular problem. The parameters evolved were q_0, β, γ, ρ_{local}, ρ_{global}. In the algorithm each ant maintained its own set of parameters, and a second pheromone matrix was kept to learn appropriate values for each parameter. The

length of the paper meant that it did not go into great detail, but it did show that the values of the parameters the algorithm converged on were similar to those produced by parameters adjusted manually.

From these investigations it certainly seems possible to create algorithms to evolve parameters, both on-line and offline. Throughout this chapter the algorithms that have been related all describe a robustness in the range of values that the various parameters can be set to. Along with this finding some of the papers were able to evolve parameter settings similar to those in the original research. There are outstanding questions such as how do parameters interact, and which parameters actually need dynamic settings (most efforts evolve as many as possible)? At an even lower level more work needs to be done on what exactly the parameters are doing, which manifests itself in the qualitative difference between two quantities for a particular parameter. An example of this is "what is the effect of altering a parameter from 0.1 to 0.12?" If there is no effect then the domain of the parameters needs to be refined but, if there is, is it possible to quantify the change a priori?

2.4 Alternative Frameworks and Algorithms

Although the primary focus of this thesis is on Ant algorithms, a number of these algorithms have had influences from other fields of research. It is worth therefore giving a brief overview of these fields so that what makes Ant algorithms unique, but also how they are similar to other methods, can be understood. This comparison allows the possibility of using previous research from other fields to influence our understanding of Ant algorithms.

For each of the topics a brief description of what the algorithm does and a discussion on how it differs to the Ant formulation is given. In Subsection 2.4.1 another Ant-inspired framework is described, followed in Subsection 2.4.2 by an overview of Stochastic Gradient Descent, which is the basis of most hill-climbing algorithms. Other metaheuristic algorithms are then explored in Subsection 2.4.3 to Subsection 2.4.5, followed by the Cross-Entropy Method and Estimation of Distribution Algorithms in Subsections 2.4.6 and 2.4.7 respectively.

2.4.1 API

The first framework, found in [Monmarché et al., 1999; Monmarché et al., 2000], is another type of Ant algorithm and was inspired by the *Pachycondyla apicalis* ant species (API comes from the first three letters of apicalis). The behaviour of interest in this species is the way in which it forages for prey. The ants have a relatively simple, efficient strategy for foraging where individual ants hunt alone and try to uniformly cover a given area around their nest. This is achieved by performing parallel local searches on various sites with a sensitivity to successful sites. The other significant behaviour is that the colony moves periodically. In addition there is some concept of recruitment of other ants, called *tandem running*, where one ant is led to an interesting area by another ant.

The computational model of this behaviour will now be described. Let there be n agents $a_1, \ldots, a_n$, located in a search space S and these will try to minimise an evaluation function f which is defined from the search space S to the set of real numbers, $\Re$. Each point $s \in S$ is considered to be a valid solution to the considered problem. In the above papers examples are given for continuous, binary and permutation spaces, making this algorithm independent of the search space type.

To apply the algorithm to S two operators need to be defined, O_{rand}, which generates a point in S uniformly and randomly and O_{explo}, which generates a point s' in the neighbourhood of a point s. The size of the neighbourhood, used in the second operator, is parameterised with an amplitude A defined in the range $[0,1]$, which differs between each ant. This second operator can also incorporate a priori knowledge in the form of a heuristic.

The nest is placed at a point $N \in S$, the area around this point is then uniformly explored using O_{explo}. This exploration is performed for T_N iterations of n ants and then the nest is moved to the best point found since the last nest move. Around this nest each ant leaves to create p hunting sites in its memory by using O_{explo} with an amplitude set to $A_{site}(a_i)$. At each hunting site the ant uses O_{explo} to assess the quality of the site, this also has an amplitude $a_{local}(a_i)$. The quality of the site is the number of prey (better solutions) it finds around the site. Those sites for which no prey is found are forgotten after $P_{local}(a_i)$ searches.

Recruitment is achieved by selecting two agents at random, if the best site found by ant a_i is a better site than the best site found by a_j then that site is copied to a_j and its old

site is then deleted. This strategy is a form of exploitation by increasing the amount of search in successful areas.

The resulting algorithm is found in Figure 2.17.

Figure 2.17: Algorithm for API [Monmarché et al., 1999]

```
1  Choose randomly the initial nest location N;
2  while stopping criteria not reached do
3      Simulate an exploration for each ant a_i, i ∈ [1..n];
4      if a_i has less than p hunting sites in memory then
5          Create a new site around N;
6      else
7          if the previous site exploration was successful then
8              Explore this site again;
9          else
10             Select randomly and uniformly another site in memory;
11             Explore this new site;
12         endif
13     else
14         Possibly delete successful sites from memory;
15     endif
16     Perform recruitment (best site copies between two randomly selected ants);
17     if more than T_N iterations have been performed then
18         Change the nest location;
19         Reset the memories of all ants;
20     endif
21 endw
```

This algorithm is very different from the Ant algorithms previously described. It is highly dependent on the operator O_{explo} (for instance in the TSP problem 2-Opt, a relatively weak local search method, was chosen as the operator and the optimal solution was not found in the tests). It is highly flexible in terms of the types of domain it can be applied to, and consequently is more easily adapted to these domains than the ACO Metaheuristic.

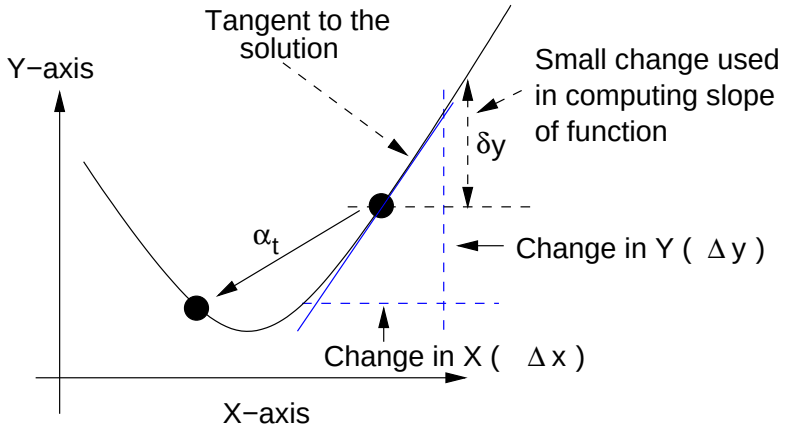


Figure 2.18: Illustration of the Equation 2.27 on a quadratic function. The figure shows that if α_t is set too large it will pass over the local minimum.

2.4.2 Stochastic Gradient Descent

Stochastic Gradient Descent is the on-line version of Gradient Descent. For this reason, the latter will be described first and then the relationship between the two will be clarified.

Gradient Descent [Gurney, 1997] is a method for finding the minimum of a particular function. The method of doing this relies on calculating the gradient of the function at the current solution point and then taking a step α_t in the downward direction. This is illustrated in Figure 2.18. By iteratively applying the formula in Equation 2.27 the local minimum will be eventually reached.

$$\Delta x \approx -\alpha_t \frac{dy}{dx} \tag{2.27}$$

In Figure 2.18 α_t was set to be too large so it passed over the local minimum. For this reason the value of α_t has to be selected with caution, normally it is reduced with respect to time, for instance a function such as e^{-t} is used. It should be clear that if the minus sign was removed from Equation 2.27 one could climb a maximum which would be called (Stochastic) Gradient Ascent.

In a continuous domain it is preferable that the function provided should be differentiable. Unfortunately when applied to a discrete problem, although one can make the relaxation of a discrete to continuous domain, it still leaves the necessary step of calculating the fitness for every solution in the space. This is rarely possible given the

nature of the problem. Therefore Stochastic Gradient Descent uses the mean of a sample to estimate the gradient of the objective function in a particular neighbourhood, or perturbation, which is why it is called stochastic.

This method has been used in [Dorigo et al., 2002b] to derive a pheromone update rule similar to that used in Ant System. However, this is where the similarities end, Stochastic Gradient Descent has no memory or population and therefore has no means of exploring the search space using anything other than hill-climbing.

2.4.3 Simulated Annealing

Simulated Annealing was first proposed in [Kirkpatrick et al., 1983], when a similarity was spotted between Combinatorial Optimisation and Statistical Mechanics, or more precisely in the way in which a metal cools and freezes into a minimum energy crystalline structure, which is called the *Annealing Process*.

The Boltzmann probability factor in Equation 2.28, is at the heart of this method and slowly collapses the space of probable solutions until convergence is achieved. One of the key properties of the algorithm is that, unlike other hill climbing algorithms, the method sometimes accepts worse solutions than its current best. This gives it a greater probability of getting itself out of local optima.

$$p = e^{-\frac{\delta f}{T}} \tag{2.28}$$

δf is the increase in the function f and T is a control parameter known as the *temperature*. The algorithm requires four items to work: a representation of possible solutions, a generator of random changes in solutions, an objective function, and an annealing schedule. The schedule contains an initial temperature and rules for lowering it as the search progresses.

This method has been highly successful in many applications and a number of variants of the algorithm exist for defining the four items in the previous paragraph. In a comparison with Ant algorithms Simulated Annealing can be classed either as a competitor or can be used as a local search in a hybrid implementation. The major difference between the two algorithms is the memory in Ant algorithms that allows it to keep a record of previous search efforts.

Figure 2.19: Simulated Annealing ([van Laarhoven, 1987])

input : A solution s

output: A solution s and its objective value $f(s)$

```
1  m ← 0;
2  repeat
3      repeat
4          Perturb(s, s');
5          Δf_ij = f(i) − f(j);
6          accept = false;
7          if Δf_ij > 0 then
8              accept = true;
9          else
10             if e^(−Δf_ij / T_m) > random[0,1] then
11                 accept = true;
12             endif
13         endif
14         if accept then
15             s ← s';
16         endif
17     until equilibrium is approached sufficiently close;
18     T_(m+1) = g(T_m) where g() is some strategy for reducing temperature;
19     m = m + 1;
20 until stop criterion = true;
21 return ( s, f(s) )
```

2.4.4 Tabu Search

Tabu Search was first published by Glover in the 1986, but has roots in the 1970's. The latest publication of Tabu Search and its associated research is [Glover and Laguna, 1997]. The technique provides approximations that are better than those provided by classical techniques and can be made to be cutting-edge with various enhancements developed in the 1990's. The theoretical aspects of Tabu Search were investigated in the early 90's by Faigle, Kern, Glover and Fox. In terms of convergence Tabu Search follows a proof similar to that devised for Simulated Annealing and it can be shown that Tabu Search should converge almost surely to the global optimal solution (Faigle and Kern, 1992). Like Simulated Annealing this search technique can be viewed either a local search procedure or as a metaheuristic.

The key to Tabu Search is its use of a *memory* to store information about the exploration process. This process relies on the definition of a *neighbourhood* about the solution i, $N(i)$. The role of the memory is to further restrict this neighbourhood to try and decrease the probability of revisiting old areas of the search space. This restricted neighbourhood is called V^*. This memory is said to make some solutions *Tabu*, hence the name of the search procedure.

The neighbourhood of a solution varies from iteration to iteration and therefore Tabu Search can be thought of as a *dynamic neighbourhood search technique*. As with the other techniques a range of notation is involved. $f(i)$ is the objective function that defines a real number for each solution in the search space. i^* is the optimal solution and therefore $f(i^*) \leq f(i)$ for every solution i in the search space S. V^* in a classical descent algorithm would normally be equal to $N(i)$ but for Tabu Search, because this neighbourhood is then refined, V^* may be a subset of $N(i)$. The choice of $N(i,k)$ and V^* is crucial to the efficiency of the algorithm; too big and the algorithm becomes slow, too small and the algorithm will converge prematurely on a local optimum.

There are four main stopping conditions for Tabu Search, which are the following:

- The neighbourhood is empty, $N(i,k+1) = \emptyset$.
- k is larger than the maximum number of iterations allowed.
- The number of iterations since the last improvement of i^* is larger than a specified number.

- Evidence can be given that an optimum solution has been obtained.

Normally the Tabu list T is made up of the most recent moves, which reduces the potential for going round in circles. This list is made up of (reversible) moves that can be applied to solution i to obtain a new solution j. In some cases it may be necessary to use a number of Tabu lists at the same time and therefore these lists are indexed by a subscript r. There are also *aspiration level conditions*, $a(i,m)$, when the Tabu list is overruled because a particular move meets some criteria that makes it irresistible, for instance greater than a threshold value $A(i,m)$.

Like other search techniques Tabu Search has methods for intensifying and diversifying the search. Intensification is achieved by modifying the objective function to penalise new moves that create solutions dissimilar to the original solution. Diversification is attained via the same means but those solutions that are similar to the original are penalised. These schemes have weights applied to them so that the process alternates between the two $\tilde{f} = f + \text{intensification} + \text{diversification}$. The full algorithm can be seen in Figure 2.20.

Figure 2.20: The Tabu Search Algorithm [Aarts and Lenstra, 1997]

1 Choose an initial solution $i \in S$. Set $i^* = i$ and $k = 0$.
2 Set $k = k+1$ and generate a subset V^* of solutions in $N(i,k)$, such that either one of the Tabu conditions $t_r(i,m) \in T_r$ is violated ($r = 1,\ldots,t$) or at least one of the aspiration conditions $a_r(i,m) \in A_r(i,m)$ holds ($r = 1,\ldots,a$).
3 Choose a best $j = i \oplus m \in V^*$ with respect to f or to the function $\tilde{f}$, and set $i = j$.
4 If $f(i) < f(i^*)$, then set $i^* = i$.
5 Update the Tabu and aspiration conditions.
6 If one of the stopping criteria is met, then stop. Else go to Step 2.

Tabu Search is a powerful local search technique that, when combined with Ant algorithms, has achieved some of the best results on Quadratic Assignment problems [Stützle, 1997] and Multi-Processor Scheduling [Ritchie, 2003]. Using Tabu Search in small, intense areas of the search space allows ACO to provide a global perspective. The largest common area of research is how to combine these two algorithm types to achieve the best results. One of the drawbacks of using Tabu Search is that it brings with it a number of parameters and design decisions, most of which are not simple to set. This therefore extends the amount of experimentation required to optimise the hybrid algorithm.

For a full discussion of the work related to Tabu Search one should consult [Glover and Laguna, 1997], which covers many of the extensions that can make Tabu Search one of the best metaheuristics.

2.4.5 Genetic Algorithms

Genetic Algorithms (GA) is an established field and a great deal of work has been done in trying to apply it to a range of applications. As with the other algorithms in this section, an exhaustive review of Genetic Algorithms and why they work is not within the scope of this thesis. In this subsection the differences between this approach and Ant algorithms will be highlighted. The Genetic Algorithm description will be based upon [Michalewicz, 1999].

Genetic Algorithms, and in fact all Evolutionary Algorithms (EA), are based on a simplified version of Evolution. Therefore, the driving force behind the algorithms takes into account a number of factors:

- Survival of the fittest, whereby those individuals that are not sufficiently adapted to their current environment perish, and their traits are therefore removed from the general population.
- Finite resource, resulting in competition and so an innate method for adapting more effectively to the environment to maximise resource use.
- Reproduction, the creation of offspring from parents where traits are inherited by the children in a non-deterministic manner from the parents.

A computational model of Evolution, for any particular problem, must have the five following components that are then combined into Figure 2.21.

- A representation for potential solutions to the problem.
- A way to create an initial population of potential solutions.
- An evaluation function, rating solutions in terms of their *fitness*.
- Operators that alter the composition of parents to generate children (common types are crossover and mutation).
- Values for various parameters that the Genetic Algorithm uses (population control, for example elite, tournament, island models; population size; probabilities

of applying genetic operators; etc).

Figure 2.21: Structure of an Evolutionary Program

```
1 t ← 0;
2 initialise P(t);
3 evaluate P(t);
4 while not termination-criteria do
5     t ← t + 1;
6     create P(t) from P(t − 1);
7     evaluate P(t);
8 endw
```

Many of the design decisions that are necessary for Genetic Algorithms also need to be made with Ant algorithms (such as problem representation, size of population/ants per iteration, and when to combine with local search). The question of which algorithm to use is generally decided by applicability to the problem in question, an example of this decision can be seen in [Keinprasit and Chongstitvatana, 2004]. In terms of representation of the problem Ant algorithms have the advantage of being easier to fit to problems where there exists an explicit graph representation. The constructive nature of the Ant algorithms removes the need to develop non-trivial operators to generate new solutions. Both algorithms have theoretical problems in common such as how to define convergence, and how long will it take to reach the optimum, if it can be reached at all.

2.4.6 The Cross-Entropy Method

As the Cross-Entropy (CE) Method is too complicated to describe succinctly, the application of it to the Travelling Salesman Problem will be described here to illustrate the main concepts involved. The description was provided in [Rubenstein and Kroese, 2004]. Behind the Cross-Entropy Method are two iterative phases:

1. Generation of a sample of random data (also known as trajectories, vectors, etc.) according to a specified random mechanism.

2. Updating the parameters of the random mechanism, typically parameters of probability density functions (pdfs), on the basis of the data, to produce a “better”

sample in the next iteration.

This strategy should be familiar as it is very similar to that of the previous Ant algorithms and, as was shown in [Dorigo et al., 2002b], one can derive similar rules for ACO from CE.

Figure 2.22: Main CE Algorithm for TSP

1. Choose an initial reference transition matrix $\hat{P}_0$, say with all off-diagonal elements equal to $\frac{1}{n-1}$. Set $t = 1$.
2. Generate a sample $X_1, \ldots, X_N$ of tours via Figure 2.23, with $P = \hat{P}_{t-1}$, and compute the sample $(1-\rho)$-quantile $\hat{\gamma}_t$ of the performances according to Figure 2.24.
3. Use the *same* sample $X_1, \ldots, X_N$ to update $\hat{P}_t$ via Equation 2.29.
4. Apply Equation 2.30 to smooth out the matrix $\hat{P}_t$.
5. **if** *for some* $t \geq d$, *say* $d = 5$, $\hat{\gamma}_t = \hat{\gamma}_{t-1} = \cdots = \hat{\gamma}_{t-d}$ **then**
   stop;
   else
   t =t +1;
   Goto Step 2;
   endif

The main algorithm, shown in Figure 2.22 provides a wrapper, the contents of which implement the two phases described above. Each iteration of these steps is indexed by a variable t, where $t = 1$ is the start. Also at the start of the algorithm the matrix $\hat{P}_0$ is initialised so that each element corresponding to a destination of an arc is given a uniform probability.

Figure 2.23: Trajectory Generation Using Node Transitions

1. Define $P^{(1)} = P$ and $X_1 = 1$. Let $k = 1$.
2. Obtain the matrix $P^{(k+1)}$ from $P^{(k)}$ by first setting the X_k-th column of $P^{(k)}$ to 0 and then normalising the rows to sum up to 1. Generate X_{k+1} from the distribution formed by the X_k-th row of $P^{(k+1)}$.
3. If $k = n-1$ then stop; otherwise set $k = k+1$ and go to step 2.
4. Evaluate the length of the tour via objective function.

In step two of the algorithm a sample is generated using the matrix $\hat{P}_0$ as its distribution. The details of this generation are given in Figure 2.23. For each column in the matrix,

the values are set to zero as the solution is constructed and their probability is shared out among the remaining possibilities. The samples are then ordered by their objective value so that the $(1-\rho)$-quantile can be selected, where ρ is generally about 0.01. This is analogous to choosing either the global or iteration best. The CE method leaves it to the choice of ρ to determine this.

Figure 2.24: Adaptive updating of γ_t

1 Let $S(X)$ be the same performance, where $X \sim f(\cdot;P_t)$. For a fixed P_t, let γ_t be a $(1-\rho)$-quantile of $S(X)$ under P_t. That is γ_t satisfies the following: $\Pr_{P_t}(S(X) \geq \gamma_t) \geq \rho$ and $\Pr_{P_t}(S(X) \leq \gamma_t) \geq 1-\rho$. A simple estimator $\hat{\gamma}_t$ of γ_t is the order statistic $\hat{\gamma}_t = S_{(\lceil(1-\rho)N\rceil)}$.

$$\hat{p}_{t,ij} = \frac{\sum_{k=1}^{N} I_{\{S(X_k)\leq\hat{\gamma}_t\}} I_{\{X_k\in\chi_{ij}\}}}{\sum_{k=1}^{N} I_{\{S(X_k)\leq\hat{\gamma}_t\}}} \tag{2.29}$$

Once the samples have been ordered, they are used in Equation 2.29 to update $\hat{P}_t$. The equation states that the estimated probability of each arc in $\hat{p}_{t+1}$, is equal to the product of whether the sample has an objective value less than or equal to $\hat{\gamma}_t$, 1 if yes and 0 otherwise, and whether the sample contains the arc ij, again a boolean test indicated by I_x which is an indicator function for the event represented by x. This is then normalised by dividing by the number of samples with an objective value less than $\hat{\gamma}_t$.

$$\begin{aligned} &\hat{v}_t = \alpha\tilde{v}_t + (1-\alpha)\hat{v}_{t-1} \\ &\text{where } 0.7 < \alpha \leq 1 \text{ and } \hat{v} \text{ is each row in } \hat{P}_t \end{aligned} \tag{2.30}$$

Having computed $\hat{P}_{t+1}$ a process of smoothing takes place, as described in Equation 2.30, that combines $\hat{P}_t$ and $\hat{P}_{t+1}$ together. This process is controlled by a parameter α, usually in the range 0.7 to 1.0. This can be seen as a process similar to ρ in Ant algorithms.

Finally the algorithm checks to see if the estimate of the optimal objective value has changed in the last d iterations, if not the algorithm stops. An alternative stopping criterion is when the maximal value in each row of $\hat{P}_t$ has not changed in the last d iterations.

One of the interesting concepts introduced is the estimation of N, the number of sample solutions taken at each iteration. In all the simulation results the value of N was of the order n^2, n being the number of cities. However, it is possible to reduce this to $n\ln(n)$ if each element has an equal probability of being chosen, this can be calculated using *Urn Models*[1] for uniform probability distributions. Currently no other probability distributions are able to be used with Urn Models so no progress has been made in this area.

2.4.7 Estimation of Distribution Algorithms

Another framework that has become a unique field is Estimation of Distribution Algorithms (EDA). The rationale behind EDAs comes from its relationship with Genetic Algorithms, therefore much of the nomenclature is similar. As has been shown, Genetic Algorithms have a large number of parameters, therefore one of the motivations behind EDAs is to remove this extra optimisation problem. The other motivation is to remove the difficulty in predicting the movements of the population throughout the lifetime of the algorithm. EDAs were first introduced by Mühlenbein and Paaßin 1996 and similar approaches were investigated by Zhigljavsky in 1991. [Larrañago and Lozano, 2002]

The basic outline algorithm for an EDA is given in Figure 2.25. The three main stopping criteria that are used are:

- when a fixed number of populations or a fixed number of different evaluated individuals are achieved,
- uniformity in the generated population, or
- no improvement with regard to the best individual obtained in the previous generation.

The most significant issue when using EDAs is the estimation of the probability distribution $p_l(x)$ once a new population has been generated. As the true joint distribution may be a very complicated and unwieldy equation it is often necessary to factorise according to a probability model, thereby simplifying the distribution.

[1] An urn problem is an idealised thought experiment in which some objects of real interest (such as atoms, people, cars, etc.) are represented as coloured balls in an urn or other container. One pretends to draw (remove) one or more balls from the urn; the goal is to determine the probability of drawing one colour or another, or some other properties. [Johnson, 1977]

Figure 2.25: Pseudo-code for EDA Approach [Larrañago and Lozano, 2002]

1 $D_0 \leftarrow$ Generate M individuals at random (initial population);
2 **repeat**
3 $\quad D^{Se}_{l-1} \leftarrow$ Select $N \leq M$ individuals from D_{l-1} according to the selection method;
4 $\quad p_l(x) = p(x|D^{Se}_{l-1}) \leftarrow$ Estimate the probability distribution of an individual being among the selected individuals;
5 $\quad D_l \leftarrow$ Sample M individuals from $p_l(x)$ (new population);
6 **until** *for* $l = 1, 2, \ldots$ *until the stopping criteria is met*;

The methods of simplification of this joint probability distribution split the field into four areas: no dependencies, bivariate dependencies, multiple dependencies and mixture models. As this is an overview, the primary algorithms will be given and a short description of the unique abilities that each area offers. It should be noted that the rest of this section is directed at discrete optimisation and that there exist EDAs to handle other domains that are beyond the focus of this thesis.

The first classification is the factorisation of $p_l(x)$ into the product of n univariate and independent probability distributions. There are three main algorithms in this class UMDA, PBIL and cGA. The one that will be described here is Population Based Incremental Learning (PBIL) introduced by Baluja in 1994. This particular algorithm has been chosen as it occasionally is chosen as a comparison to an Ant algorithm as in [Cordon et al., 2000b].

The PBIL algorithm described in Figure 2.26 has the objective of obtaining the optimum of a function defined in the binary space $\Omega = \{0,1\}^n$. In each generation the population of individuals is represented by a vector of possibilities:

$$p_l(x) = (p_l(x_1), \ldots, p_l(x_i), \ldots, p_l(x_n)),$$

where $p_l(i)$ is the probability of obtaining the i^{th} component in D_l.

The next most sophisticated group of methods is bivariate dependencies, which are dependencies between two of the variables. Again there are three algorithms that fall into this category: MIMIC, COMIT and BMDA. This classification is a special case of the third group which deals with multiple dependencies. This group contains EcGA, FDA, PADA, EBNA*, BOA, and LFDA. All of these algorithms follow a similar pattern of creating a tree of dependencies to a particular depth, generating a new population and

Figure 2.26: Pseudo-code for the PBIL algorithm

```
1  Obtain an initial probability vector p_0(x);
2  while no convergence do
3      Using p_l(x) obtain M individuals: x^l_1,...,x^l_k,...,x^l_M;
4      Evaluate and rank x^l_1,...,x^l_k,...,x^l_M;
5      Select the N (N ≤ M) best individuals: x^l_{1:M},...,x^l_{k:M},...,x^l_{N:M};
6      Update the probability vector p_{l+1}(x) = (p_{l+1}(x_1),...,p_{l+1}(x_n));
7      for i = 1,...,n do
8          p_{l+1}(x_i) = (1−α) p_l(x_i) + α (1/N) Σ_{k=1}^{N} x^l_{i,k:M}
9      endfor
10 endw
```

then starting again. The main point of difference is either the way they build the tree, for instance using Bayesian Network, or in the way these distributions are evaluated, for instance using the Kullbeck-Leibler distance rather than the Bayesian Dirichlet equivalence. Suffice to say that if one required the algorithm to learn complex structure it is possible for it to do so. The disadvantage with many of these methods is the time they take to calculate the distributions and also that they are limited by the number of samples that the distribution is induced from.

One solution to these problems comes in the form of the final category, the Mixture Model approach. This approach assumes that the data can be clustered together by k Gaussian distributions. This removes the need to induce a dependency structure from the population and one can quickly adjust the set of Gaussians to the current population. However, the disadvantage with this method is that of selecting a value for k for a particular problem. There are methods available to do this, if necessary, but again it introduces another optimisation problem in addition to the original one.

2.5 Conclusions

To conclude this tour of current literature a short critical analysis is given of the material covered. There are three areas of analysis that require consideration:

- *Experimental Issues*: Although a great deal of work has been done, the quality of experimentation has been difficult to ascertain in the papers that have been

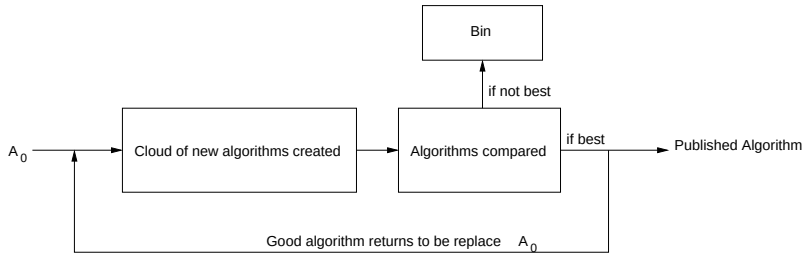


Figure 2.27: Life-cycle of an Ant algorithm

collated for this review. In this subsection some criticisms are made and some suggestions given.

- *Reporting of Results*: A brief discussion is given of the quality of the reporting of results and the frequency of interesting results.
- *Open Questions*: Some open questions are raised that require further study.

2.5.1 Experimental Issues

As one considers the various algorithms in this field one of the realisations that comes across is that no matter what is changed it always seems to improve the basic Ant System algorithm. This is either because Ant System is a particularly badly formulated algorithm, or because experiments are not run using comparable algorithms. The actual reason for these results is likely to be a combination of these two factors. More effort goes into the performance of the new technique than optimising the control algorithm, hence producing these 'successful' results.

It is also clear that different problems and their constraints warrant different memory models and rules for manipulation of these models. This is exemplified by the Relative Pheromone Evaluation Rule introduced in [Merkle and Middendorf, 2001, 2002a]. In this respect what appears to be happening is that a researcher may pick a problem domain d, apply Ant System to it and then create a new algorithm which is more optimised to the particular domain. Comparison of these results will invariably result in conclusions in favour of the new algorithm, therefore this result is not very informative. This *algorithm life-cycle* is illustrated in Figure 2.27. It would be of more use if the algorithms were to be compared on problems that neither was optimised for to find which algorithm worked best on which instances.

Another criticism of the comparisons reported in many of the papers is that when the

algorithms were hybridised the authors often stated what the Ant algorithm did without the associated method, but it was not stated how the other search method performed without the Ant algorithm. Few papers showed that they had explored this possibility and therefore failed to show that it was the Ant algorithm that was making the difference. One presumes that they knew the local search would not get as high quality results but the authors failed to relate this and thus the paper loses its authority.

It is worthy of note that when dealing with all the described algorithms there exist two optimisation problems. The first is the one the researcher is trying to solve and the second is the meta-problem that is to optimise the parameters of the algorithm and in the case of hybrids to optimise the choice and mixture of algorithms. For single run problems it is sufficient to get a reasonable compromise in the parameters of the algorithm. Unfortunately this is a double-edged sword and if the parameters are assumed to be robust then no experimentation will be done to provide the evidence for this over a variety of domains. More information would be gained by reporting the parameters that did not work rather as well as those that provided the best solutions.

2.5.2 Reporting Issues

Linked to the above point on experimentation is a similar deficiency in the reporting of results. The first and one of the most important points that became evident as this literature review was collated was the lack of rigorous results. This became apparent as every paper cited the speed of the algorithms on various problems but did not give the necessary information about what did not work, what did not produce quality solutions, and what parameters were used. Out of the 225 papers reviewed approximately 26% attempted to quote their parameters, while only 73% of these, 21% of the entire group, gave evidence that they had even tried other variations and gave reasons for their choices. This demonstrates a significant problem within this field. It will therefore be one of the goals of this thesis to explore in a scientific way the knowledge input and manipulation occurring in these algorithms.

Another minor problem that becomes apparent only when reading a large number of papers is overloading of variable names. Although the situation has improved, the notation used is often influenced by the algorithm one starts off with. An example of this confusion is that α is used in the Ant System as the influence of the pheromone matrix whereas in Ant-Q it is used in the pheromone update rule.

2.5.3 Open Questions

There have been a number of algorithms in this chapter that have been described that can solve the same problem to approximately the same degree. The problem with the studies that have taken place is the lack of recording of the quality-effort trade-off that each algorithm makes. While Ant algorithms do improve the best known result for a great deal of problems it is not clear exactly what the trade-off is in this respect. For instance have the Ant algorithms been given longer? Or are they at an advantage in some way to the comparison algorithms? No paper has investigated this trade-off in a way that clarifies the decisions that would have to be made when choosing a particular algorithm to solve a given problem.

Related to this idea of a quality-effort trade-off is the evidence that increasing the number of ants above a certain point no longer gives a significant benefit to the performance of the algorithm. This may be a valid point, but there has not been any work done to try and find out why, and when, this occurs in order to optimise this apparent trade-off. This trade-off is looked at in Sections 5.6 for algorithms without local search and in Section 7.5 with local search. Many of the other algorithms described here have similar problems, thus more research is likely to benefit the wider research community.

Although it is not necessary for the discussion of new ideas for models to be implemented, it is desirable when claims are made about properties of a particular algorithm to show implementations. In this regard it is necessary to provide an implementation of the Graph-based Ant System, which is to date the best theoretical model of Ant algorithms. This would demonstrate a number of points. Firstly, it would demonstrate that the implementation does show, in a practical environment, the characteristics aspired to by the model. Secondly, it would show how the implementation of the model compares to popular algorithms like Ant System and the Max-Min Ant System in terms of the effects of parameters. This would prove or disprove the reliability of generalising results from the Graph-based Ant System to these other more practical implementations. This is the core question studied in Chapter 5 when the Graph-based Ant System is compared to Ant System and the Max-Min Ant System.

Another issue that has arisen is the requirement to shake or smooth the pheromone matrix. This involves spreading some of the pheromone received for a good solution not only to the items in a particular solution but also to the local neighbourhood. The question remains as to when this is a good idea and what is the best method to achieve

it.

On the related topic of exploration, in Subsection 2.2.9 a need for a pruning method was mentioned. This would be a useful function to achieve automatically and would aid in the scaling of ACO to larger problems. However, the question of how to achieve this still requires further investigation.

The information given to the algorithms via heuristics in all the work seen in this chapter has been made as valid as the researcher has been able to ensure. An interesting research question is how robust are the Ant algorithms if the heuristics and other information they are given is not completely accurate, even to the extent of being misleading. This is the question that Chapter 6 begins to investigate.

Although many of the applications of Ant algorithms have achieved successful results, this success has normally been achieved in combination with some type of local search method. It appears from the research that there has been no study on what the effect of using a local search method is on behaviour of Ant algorithms. This is a topic that is studied in more depth in Chapter 8.

The final point is one introduced in the algorithm FANT, which focused on the method of reinforcement of solutions. In the majority of algorithms the pheromone matrix is updated proportionally to the quality of a solution, but in FANT a points system is used. This prompts the question: what is the best method of reinforcement? A related issue that has been studied in some other papers is the idea of negative reinforcement. The relationship between the use of positive and negative reinforcement [Montgomery and Randall, 2002] is an interesting question that has still to be investigated.

2.6 Summary

This concludes the literature review. In this chapter the history of the field of Ant algorithms has been described, followed by a review of the some competing algorithms. The chapter ended with a critical analysis of the papers, citing issues with the research and demonstrating that there is a definite requirement for more research in certain areas in particular.

Chapter 3

Experimental Methodology

In this chapter the experimental methodology that will be followed is described. This methodology will be used in Chapters 4 onwards. Most of this chapter outlines normal scientific method and is included as clarification. In Section 3.1 the difference between empirical and rational methods is discussed. This difference is an important one in this thesis, as one of its objectives is to take a theoretical model and implement it. This implementation bridges the gap between these two methods, therefore it is important to understand what is involved. This is followed by a discussion of the methods and questions that are of interest in investigations of this type.

Section 3.2 discusses the details of the application created for the experiments in this thesis. Subsection 3.2.2 includes a discussion on the debate over Random Number Generators, which are essential to probabilistic algorithms yet can vary greatly in quality. In Section 3.3 the problem sets are scrutinised, the debate over benchmarks versus random problem sets is discussed, and the algorithms for generating random problems are given. Section 3.4 gives details of the basic statistics that this thesis makes use of to illustrate and reinforce its claims. In this section the concept of hypothesis testing is described and the parametric and non-parametric tests used are given. The chapter concludes with a summary.

3.1 Experimental Process

Empiricism is a philosophy where by the knowledge one considers as true is inferred from data collected in the world. The opposite of empiricism is rationalism; this is the

deduction of conclusions from assumed axioms using proofs. Both have their place in scientific research, although in general a conclusion is considered stronger if derived from theory rather than from empirical evidence. Normally algorithms that are probabilistic in nature are studied empirically. However, theoretical work is possible when dealing with the probabilistic algorithms described in Chapter 2, although the models that are used are normally simplifications of the implementations. This can lead to a disparity between the results of the implementation and those predicted by the model. One advantage of empirical investigations is that they can go on to investigate algorithms that do not currently come under the auspices of the model. This thesis is an empirical investigation, therefore in this chapter the methodology is described.

An empirical investigation consists of a set of experiments. An *experiment* is a set of tests run under controlled conditions for a specific purpose: to demonstrate a known truth, to check the validity of a hypothesis, or to examine the performance of something new. A *factor* is any controllable variable in an experiment that influences the outcome or result of an experiment. [Barr et al., 1995]

The experimental process can be defined using the five steps in the list below.

1. Define the goals of the experiment.
2. Choose measures of performance and factors to explore.
3. Design and execute the experiment.
4. Analyse the data and draw conclusions.
5. Report the experiment's results.

The first step is a statement of the questions to be answered and the reasons that experimentation is required. The goals of an investigation can help guide and focus the investigator and help to identify the type of results to seek, the hypotheses to test, the factors to explore, the measures to use, and the data needed to support all of the above.

After the goals have been defined one should have a clear idea of what measurements and factors will produce relevant data. There are two types of variables to consider when formulating measures: *dependent* variables are the performance measures or results; and *independent* variables are the variables one cannot control, or are static, such as the problem, the algorithm and the test environmental factors. Both sets must be chosen with the identified goals in mind. Some measurements that should be con-

sidered when studying a particular algorithm are:

- What is the quality of solution achieved?
- What is the time taken to determine the best solution?
- How robust is the method?

Sometimes raw measurements do not help to illuminate an answer to the particular goal and it may be necessary to combine measurements in a statistic. One example of this is when comparing two algorithms. To determine how much work a particular algorithm does to achieve a certain increase in quality the statistic in Equation 3.1 could be used.

$$r_{0.05} = \frac{\text{time to within 5\% of best}}{\text{time to best found}} \tag{3.1}$$

Having completed the first two steps, next the experiment itself must be defined. A good experiment meets a number of requirements, while minimising bias. *Bias* is the term used for any factor that might influence the outcome of an experiment. Bias is introduced in many ways and it can be difficult to rule out all forms. Some common biases might be using an inadequate Random Number Generator, or the researcher only using certain parameters. An experiment should also fulfil a basic set of criteria: it should achieve the experimental goals, clearly demonstrate the performance of the tested process, uncover the reasons for performance, have a justifiable rationale, generate supporting conclusions and be reproducible. The final criterion is very important and is sometimes forgotten when it comes to publishing papers.

The final step in the experimental process is the analysis of results. This is made up of statistical tests and visualisations of the data. These are then interpreted, which leads to conclusions and inferences that may have statistical and practical significance. The report should also discuss any implications of the conclusions and make relevant recommendations for researchers wishing to continue the work.

Having described the basic process that will be performed to collect the results for this thesis the rest of this chapter is devoted to some of the most important aspects of this process. The first section explains the implementation and how it has been developed to help to minimise the bias for one particular algorithm or domain. Following this the problem sets that have been chosen will be introduced, after which a summary of basic statistical tests that will be used to analyse the results generated by the implementation will be given.

3.2 The SI-Testbed Application

This section describes the application that was created to investigate the questions posed in this thesis. The application was named SI-Testbed and has an architecture that allows as much code as possible to be shared between the algorithms under investigation. In Subsection 3.2.1 technical aspects of the application are explained and the design decisions are discussed. Following this is a discussion of Random Number Generators, a crucial component of any application of this sort.

3.2.1 The Implementation

Due to the similar origin of the algorithms being studied, it seemed sensible to create a piece of software that would enable aspects of the algorithm to be plugged in and out at will and be able to run on multiple problem domains. SI-Testbed was created to fulfil this need. The idea was to have as much common code as possible to get reliable comparisons of the various algorithms. This is as opposed to previous implementations that are separate applications and so do not share code, thereby making comparison more difficult. This is especially true of algorithms as similar as the ones that are used in this thesis.

Figure 3.1 illustrates the structure of the testbed at a suitable level to highlight the modularity. Each run begins with the creation of a *factory* that contains all the information required to run an instance of a particular algorithm on a problem. Therefore there exists a factory for each problem type in the testbed.

Each factory builds an instance of a *search algorithm* for a particular *problem* instance. Problems are split into maximisation and minimisation problems. These problem classes contain the solution structure and all the required information to read and write the various problem specific information.

In the real application there are two different implementations alongside one another. One is for Constraint Satisfaction (CSP) style problems and the other for Permutation style problems. This is an important distinction as some of the data structures required differ between the two problem sets. The most significant example of this is a semantic difference in the meaning of a solution. A solution π is stored as a list of numbers. In a CSP the meaning is that the variable represented by position i is assigned a value

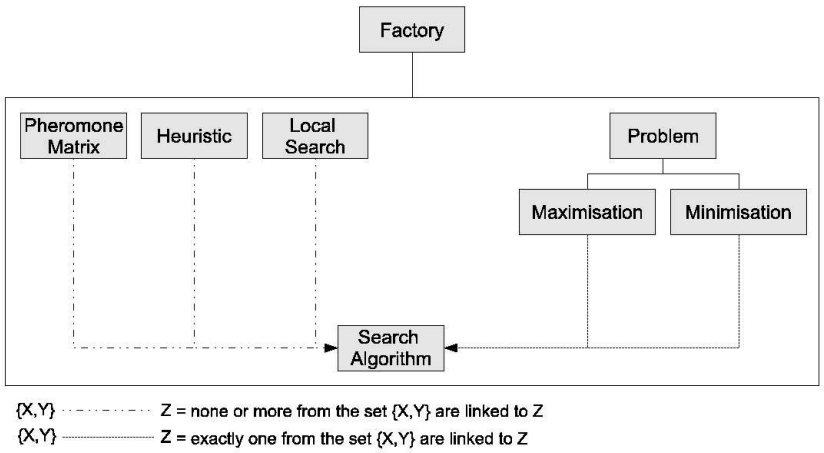


Figure 3.1: A basic representation of the SI-Testbed structure.

represented by $\pi(i)$ that may be the same as another value $\pi(j)$ at position j where $j \neq i$, but for a Permutation problem the values must all be different and are interpreted in a different way by many of the classes.

A number of structures are required by the search algorithm to create a valid instance of the problem. These are *heuristic*, *local search* and *pheromone matrix*. These represent the various parts of the algorithms that are being studied in this thesis, which may contain none or more of these structures.

This modular architecture enables a new algorithm to be implemented rapidly and tested with a range of attributes. Obviously not all attributes are always required and when specifying the options to the testbed certain combinations are checked and ruled out. One example of this might be trying to use the TSP 2-Opt local search on the QAP problem.

The common code approach means that common classes can be used to collect statistics on performance and features such as Random Number Generators, that can play a significant role in an algorithm's performance, are kept constant. This can play an important role when trying to control external variables and aid consistency in the scientific methodology.

The application was created using C++ making use of the object orientated nature of this language. Java would now also be an acceptable language, its maturity and speed are now comparable with C++, but when this investigation began this was not the case and so these disadvantages were the deciding factor in choosing C++.

There are disadvantages to this approach. The main one is that because the code is not optimised for a particular algorithm it is slightly slower, but this extra overhead is not large enough to pose a problem for performing large experiments. It is also the case that it may not be optimised for a particular problem, an example of this would be between S-TSP and A-TSP. There are optimisations one can make when programming a S-TSP solver that cannot be generalised to the A-TSP. However, as the TSPLIB was the underlying library at the time of writing, the A-TSP definition was the one the code was written for. This is most noticeable in large problems. However, as this application is not designed to be a dedicated solver for TSP problems this is not considered a major disadvantage. It could also be re-engineered if necessary.

To complete the numerous experiments that were performed two Beowolf clusters were used, one of 64 and another of 16 machines. In addition up to a further 150 machines were used at times around the Informatics Department. The machines were all of similar specification, 2GHz Pentium 4 with approximately 512MB of RAM, although none of the runs required this much memory.

3.2.2 Random Number Generators

All probabilistic algorithms, including Ant algorithms, require a random number generator. The choice of random number generator is important as it may introduce bias that is very difficult to detect. Thus it is worth discussing the various common methods used to select a reputable source of random numbers.

There are two types of sources for random numbers that can be used in probabilistic algorithms. Firstly, a source of real random numbers, a Random Number Generator (RNG), could be used. The problem with this for scientific work is that the results are not repeatable, therefore the majority of researchers use Pseudo-Random Number Generators (PRNGs). These take a seed, which is normally a very large integer, and using a deterministic algorithm produce seemingly random numbers. This algorithm can vary, thus there are good generators and bad generators.

The definition of *randomness* that is used is the following:

Definition 3.2.1 *Given a stream of bits (ones and zeros) from a generator, the probability of getting a one or zero should converge on a half (a virtual coin flip).*

This infers that each bit of output should be independent of the previous bits and there-

fore cannot be predicted. This ideal of an unbiased coin-flip serves as the benchmark when comparing and evaluating random number generators, whether real or pseudo-random.

Apart from that the sequence is generated from a single large number, there are two further conditions that are placed on the pseudo-random number generator. Firstly that one should not be able to predict the next bit output even if one is given the previous outputs. Secondly, the seed should not be predictable from a sequence of bits from the generator. These are called *forward* and *backward predictability* respectively.

Ironically many of today's Pseudo-Random Number Generators (PRNG) are more random than random number sources in terms of the statistical tests used. This is because many of the algorithms are designed to remove correlations that are detected by various statistical tests, and also the growing recognition of the fact that many random number sources are more predictable than previously thought, such as keyboard input which is a source that word completion tools rely on.

Given this background, what are the options for generating a stream of millions of random numbers? As C++ includes a PRNG function called rand(), this could be used as the source. The rand() function is a Linear Congruential Generator. The general formula for these can be seen in Equation 3.2. The values a,c and m are pre-selected constants and it is this that makes this technique risky to use. The quality of this function varies between C++ distributions, although the minimal standard PRNG has the values a = 16807, c = 0, m = 2147483647 set by Park and Miller in 1988 [Park and Miller, 1988].

> Historically, there has also been many complaints that the LCG constants used by some implementations were poor. So the real problem is: you cannot rely on rand() producing good quality random numbers. [Vecerina, 2004]

$$I_k = (aI_{k-1} + c) \bmod m \tag{3.2}$$

The current widely acclaimed PRNG is the Mersenne Twister Generator developed by Makoto Matsumoto and Takuji Nishimura in 1996/97. The period before repetition starts is $2^{19937-1}$, a very long time. One important factor is that it is very fast, it can generate output four times faster than the ANSI-C rand() function mentioned above. The code that is used in the test-suite is that written in C by [Matsumoto and Nishimura,

1998], which can be consulted for a more detailed explanation of the algorithm used.

3.3 Domains

When performing an empirical investigation of an algorithm there is significant debate over the merits of using benchmark and randomly generated problems. As is argued in [Hooker, 1996], the scientific process requires the selection of problems that will highlight a particular feature of an algorithm while keeping everything else constant. This is the purpose of randomly generated problems, that are generated to show some aspect of an algorithm and not necessarily generated to be realistic.

The role of benchmark problems should be to allow researchers to compare the performance of their algorithms in a standard way. Benchmarks are usually biased toward the motives of the researcher who compiled the set, therefore it is important to know which are random and which imitate real problems. When benchmarks are used solely to compare algorithms it may be that the algorithm in question has been optimised for these problems alone. This results in a highly specialised algorithm. For this reason it is preferable to have an unseen set of problems on which to test the algorithms besides the benchmark problems.

For these reasons, in this thesis, in addition to the use of benchmark problems, random unseen problems will be used, especially when testing hypotheses. This process will capture the advantages of both problem types and allow comparisons of performance with other papers. To make the most of the problems that are in the various libraries, all exploratory experiments will be performed only on the TSP domain. The QAP and Talent Scheduling sets have only been used for the non-exploratory experiments, which are those that appear in this thesis. In this way any bias due to favourable problems being selected is minimised.

In this section the problem domains that will be used to investigate the behaviour of the algorithms will be described. Included with the description of each problem will be some of the more common local search techniques and heuristics used. The problems described are formally solved when the final solution is proved to be optimal but, as incomplete algorithms are being investigated, the approximate solution will be the best solution found per run, called the Global Best (GB).

The problems that will be used are the Travelling Salesman Problem (TSP), Quadratic

Assignment Problem (QAP) and Talent Scheduling Problem (TS). The justification for using these problems is one of ease of comparisons. Using the first two problems opens up the possibility of replicating results already published, and highlighting any differences in particular algorithms. Both provide a generally accepted common set of problems for illustrating an algorithm's performance characteristics. The third problem is a mixture of a number of pattern sequencing problems. This set is given to provide some diversity to the experiments.

In this chapter there will not be any specific instances of problems given. The reason for this is that the specific problems used depend on the particular experiment being run. General tests that are run on all benchmarks, for instance on judging the complexity of a benchmark, will be given in an appendix that will be referenced at the relevant point.

3.3.1 Travelling Salesman Problem

Before formalising the Travelling Salesman Problem (TSP) problem, the definition of a *Hamiltonian Circuit* will be defined. Finding a Hamiltonian Circuit in an arbitrary graph is another NP-Hard problem related to the TSP, which is why it is assumed that the graphs are fully connected.

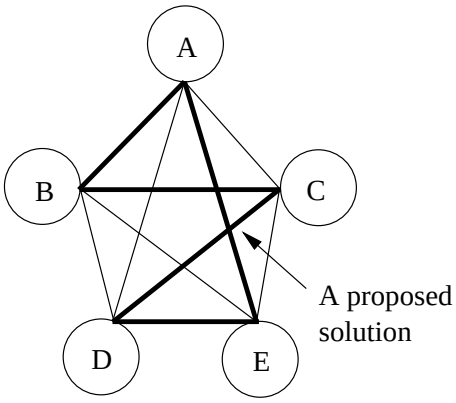
Definition 3.3.1 *A* Hamiltonian Circuit *(*cycle*) in a directed (undirected) graph is a circuit (cycle) which visits each vertex at most once.*

In relation to the TSP, Hamiltonian Circuits will be known as *tours*. The mathematical formalism of the Asymmetric Travelling Salesman Problem (A-TSP) is as follows:

Definition 3.3.2 *For a given* cost matrix *(C) minimise* $(c_{1i_1} + c_{2i_2} + \cdots + c_{ni_n})$ *where* $(i_1, i_2, \ldots, i_n)$ *is a permutation of* $(1, 2, \ldots, n)$ *which is not a product of two smaller permutations. Let* $G = (V, A, c)$ *denote the complete directed network on vertex set* $\{1, 2, \ldots, n\}$ *with the length of (or cost of traversing) arc* ij *being* c_{ij}*. The problem therefore can clearly be stated as follows: Find a minimal length Hamiltonian circuit of G. [Boffey, 1982]*

In this definition the following will be assumed for each cost $(c_i j)$ in the matrix C:

1. $c_{ii} = \infty, i = 1, 2, \ldots, n$
2. $c_{ij} \geq 0, 1 \leq i \neq j \leq n.$



Cost Matrix

inf	10	6	1	9
13	inf	3	12	6
13	8	inf	9	6
8	16	18	inf	12
25	20	4	13	inf

Figure 3.2: An example of a 5-city S-TSP problem.

The first assumption means that no graph can have loops, which are arcs that have an endpoint equal to its start point. The second states each one has a cost greater than or equal to zero, which can be assumed without loss of generality (for proof see Theorem 7.1 in [Boffey, 1982]).

There is a special type of TSP where $c_{ij} = c_{ji}$ called Symmetric Travelling Salesman Problems (S-TSP). An example of a S-TSP problem is illustrated in Figure 3.2. It is clear from this that the number of paths through a S-TSP graph is $\frac{(n-1)!}{2}$ and for an A-TSP is $(n-1)!$.

The TSP problem is easy to state but has been studied in great detail and continues to be so due to the fact that it is as difficult as any other NP-Complete problem. Although the problem is idealised it does have some important applications, for instance collection and delivery problems and some scheduling problems can be modelled as TSPs. Another reason for using this domain is that there are a large number of benchmark problems that have been compiled into a library called TSPLIB [Reinelt, 1995]. With such a wealth of benchmarks comes a multitude of prior results from various algorithms, thus it is for this reason that the TSP will be used.

Before looking at heuristics and local search techniques it is interesting to explain the main lower bound estimate, the Held-Karp bound [Williamson, 1989]. However, before this, a few more definitions are needed to understand the bound calculation. The first is repeated from Chapter 1.2 for clarification.

Definition 3.3.3 *A graph G, whether directed or not, is* connected *if there is a chain between every pair of vertices of G. If G is not connected then it can be split into*

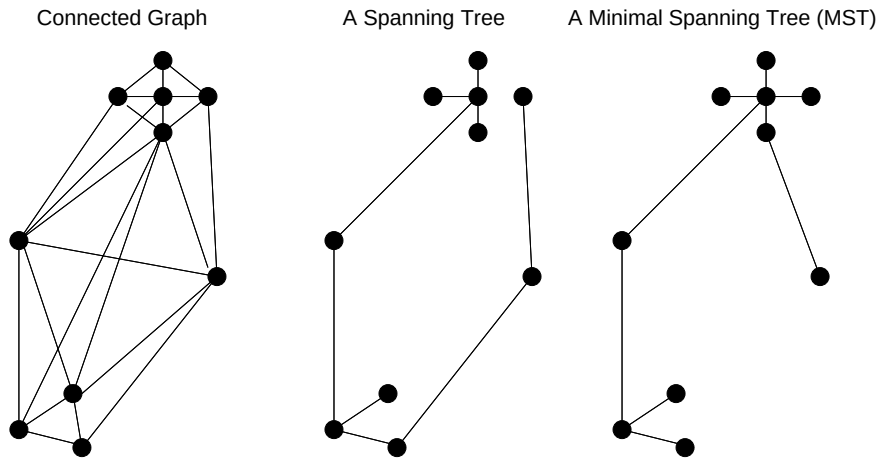


Figure 3.3: An example of a connected graph, a spanning tree and a minimal spanning tree.

components *each of which is a maximal connected graph contained in G.*

Definition 3.3.4 *If G is a connected graph on vertex set V. A* spanning tree T *is a tree, also with vertex set V, whose only edges correspond to edges (or perhaps arcs) of G.*

Definition 3.3.5 *A* minimal spanning tree *(or MST) of a network is a spanning tree with minimal weight.*

To clarify these definitions, illustrations of all three are given in Figure 3.3. An interesting algorithm to mention is one used to find the Minimal Spanning Tree (MST) of a complete network (V,E,w). This is shown in Figure 3.4 and is given to illustrate the steps required to calculate the bound.

The Held-Karp bound has proved popular for two reasons. The first is due to its accuracy when used in practice; it has been calculated on occasion to be within 99.5% of the cost of the optimal solution. The second reason is that it has been used in many complicated heuristic algorithms for the TSP, therefore demonstrating its worth. The lower bound is calculated in the symmetric case by finding an optimally weighted 1-tree, which is a spanning tree on nodes $\{2,\ldots,n\}$ plus two edges incident to node 1. This single cycle is a relaxed tour, and a minimal 1-tree can be found by calculating the MST of $\{2,\ldots,n\}$ and taking the two lowest cost edges incident to node 1. This can be summarized in the linear program in Equation 3.3.

Figure 3.4: Algorithm to find the minimal spanning tree of a network

input : Network (V,E,w)

output: Tree Structure representing minimal spanning tree

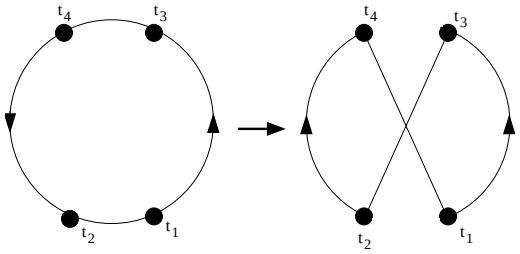
```
1  Set G_1 to be the graph with vertex set V and no edges;
2  while i < n do
3      Find a minimal weight edge xy of G_i;
4      Set w(xy) = ∞;
5      if G_i ∩ xy does not form a cycle then
6          G_{i+1} = G_i + xy ;
7          i++;
8      endif
9  endw
10 return G_n
```

$$\begin{array}{lll} \text{minimise} & \sum_{1\leq i<j\leq n} c_{ij}x_{ij} & \\ \text{subject to} & \sum_{j>i} x_{ij} + \sum_{j<i} x_{ji} = 2 & i = 1,2,\ldots,n \\ & \sum_{i\in S, j\in S, i<j} x_{ij} \leq |S| - 1 & \text{for any proper subset } S \subset V \\ & x_{ij} \leq 0 & 1 \leq i < j \leq n \\ & x_{ij} \geq 0 & 1 \leq i < j \leq n \end{array} \tag{3.3}$$

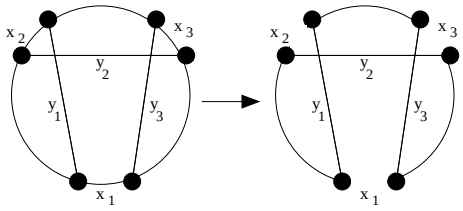
$x_{ij} \in \Re$ denotes the amount of edge ij in the relaxed tour solution. If this variable was limited to 0 or 1, the relaxed tour would be an actual tour. The asymmetric case is similar but more complex.

Although it is accurate this bound is relatively computationally intensive, therefore the main heuristic for the TSP used in more complicated algorithms is $\frac{1}{c_{ij}}$, which is referred to as the Nearest Neighbour heuristic. This heuristic directs the algorithm to pick the closest city to move to if faced with a choice between cities. The greedy algorithm using this rule is $O(n^2 log_2(n))$ and keeps within 15-20% of the Held-Karp lower bound [Nilsson, 2003]. In a non-greedy algorithm this heuristic can be used to *guide* the search to profitable areas of the solution space.

The Lin-Kernighan local search algorithm for the TSP is a very powerful local search that, when combined with Branch-and-Bound, can solve the largest TSPs. An effective implementation was written by Keld Helsgaun, detailed in [Helsgaun, 1998]. The Lin-Kernighan algorithm is a deterministic way of choosing a parameter λ such that one



(a) 2-Opt Move



(b) 3-Opt Move

Figure 3.5: TSP local search moves for $\lambda = 2$ and $\lambda = 3$ ([Helsgaun, 1998])

can replace λ edges in a solution to achieve a smaller route. The full Lin-Kernighan algorithm will not be used in these experiments because it solves the benchmark problems too easily by itself, meaning that little would be learnt from the experiments. However, a number of specific versions will be used. By limiting λ to two or three the local search is still powerful enough to locate local optima, but not so powerful that it could solve a number of benchmark problems alone.

The three local search procedures that will be used in the experiments in this thesis will be implementations of the Lin-Kernighan algorithm but for specific λ values. The 2-Opt and 3-Opt procedures perform moves as illustrated in Figure 3.5. The third, called 2h-Opt is a special version of 2-Opt that takes into account that one is on a globe and is used for "real-world" TSP problems. All three procedures were adapted from the implementation ACOTSP [Stützle, 2004].

The pseudo-code for the 2-Opt local search method is given in Figure 3.6. The algorithm assumes the following functions exist. The other two local search methods embody much of this algorithm.

- *copy(solution)* : Copies the current solution and returns the copied object.
- *generate_random_permutation(start, end)* : Generates a random permutation of

the sequence of numbers from *start* to *end*.

- *getSuccessor(solution, city)* : Gets the successor of *city* in *solution*.
- *getPredecessor(solution, city)* : Gets the predecessor of *city* in *solution*.
- *search_for_nearest_neighbour(successor, city, neighbour)* : Searches for the nearest neighbour of *city* within a radius bounded by the current distance of *successor* from *city* and puts the result in *neighbour*, returns true if a valid neighbour is found.
- *alter_solution(solution, city, oldneighbour, neighbour)* : Changes *solution* so that *neighbour* replaces *oldneighbour* next to *city*, and mends the tour.

Figure 3.6: 2-Opt Local Search Method

input : A solution s of size n

output: A locally optimum solution

```
1  s' ⟵ copy(s);
2  boolean flag_improved ⟵ true;
3  rv ⟵ generate_random_permutation(1, n);
4  while flag_improved do
5      foreach city in rv do
6          successor ⟵ getSuccessor(s', city);
7          predecessor ⟵ getPredecessor(s', city);
8          flag_improve ⟵ false;
9          if search_for_nearest_neighbour(successor, city, neighbour) then
10             alter_solution(s', city, successor, neighbour);
11             flag_improved ⟵ true;
12         else if search_for_nearest_neighbour(predecessor, city, neighbour) then
13             alter_solution(s', city, predecessor, neighbour);
14             flag_improved ⟵ true;
15         endif
16     endfch
17 endw
18 s ⟵ copy(s');
19 return s;
```

No Heuristic	2-Opt	2h-Opt	3-Opt
Number of optimums found	21	22	48
Percentage of optimums found	28.4%	29.7%	64.9%
Largest Problem Solved (cities)	105	159	264
With Heuristic	**2-Opt**	**2h-Opt**	**3-Opt**
Number of optimums found	24	31	49
Percentage of optimums found	32.4%	41.9%	66.2%
Largest Problem Solved (cities)	159	159	264

Table 3.1: Table summarising the results for local search methods applied to the TSPLIB data set.

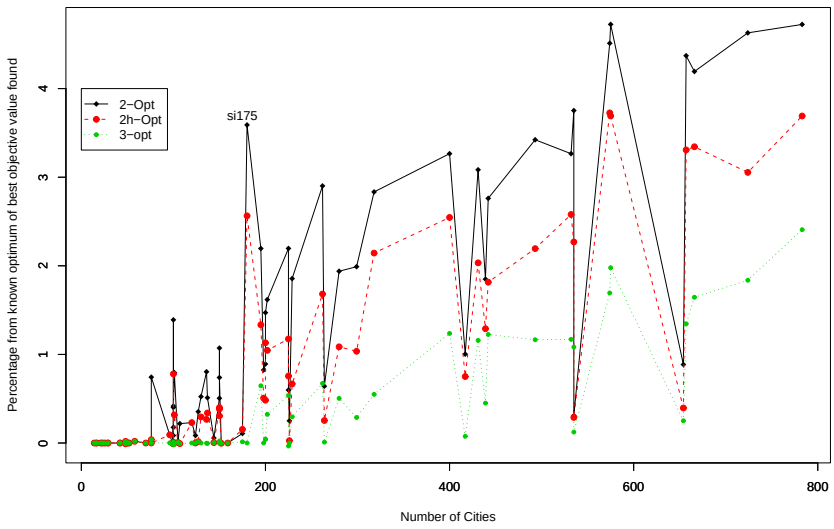
3.3.1.1 A Brief Study of 2-Opt, 2h-Opt and 3-Opt for the TSP

To investigate how easy it was to solve the various problems in TSPLIB the first experiment that was performed was to see how well the three local search methods did by themselves. Each local search was performed on 74 problems from a size of 14 up to 783 cities. Above this the problems' optima are assumed to be out of reach by local search alone within the given constraints (an assumption based on previous exploratory investigations). Ten runs of each algorithm were performed and the local search was given a randomly generated solution to enhance each iteration. Each run was given 1000 iterations to try and reach the optimum solution.

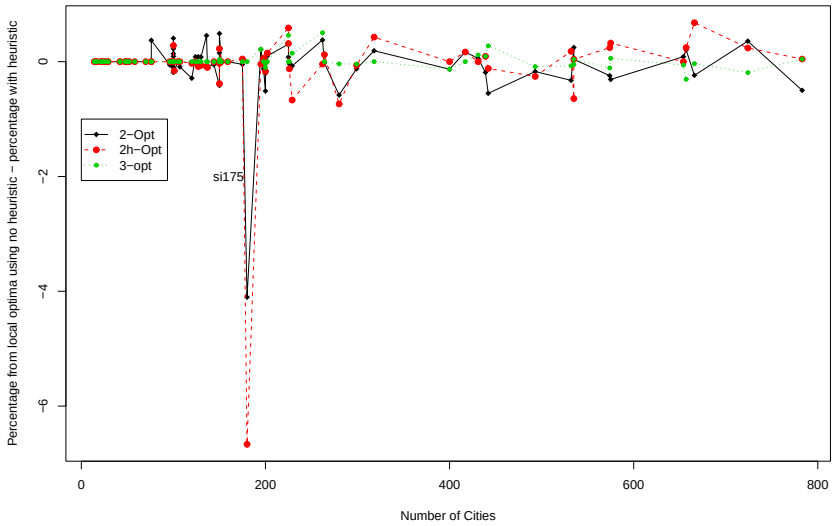
The results are summarised in Table 3.1. Unsurprisingly the more comprehensive 3-Opt local search method performed the best. Looking at the spread of the results in Figure 3.7(a) one can see that the local search methods all perform within 5% of the known optimum of each problem, although this distance increases with city size.

Next, to test the quality of the Nearest Neighbour heuristic, all of the experimental conditions remained unchanged except that, instead of receiving a uniformly random solution, the solution was biased proportionally by the heuristic. The results from this are summarised in the bottom of Table 3.1.

Table 3.1 shows that the heuristic, used in this probabilistic manner, does appear to help the local search to achieve more optima than when the local search was used alone. From Figure 3.7(b) one can see that there is only a small difference of -2 to +2% in the minimums achieved. However, for si175 it appears the heuristic hindered the ability of the local search to find good quality solutions.



(a) Graph showing the percentage error of three local search methods from the known optima when no heuristic is used.



(b) Graph showing the minimum difference between the percentage from the known optimum for each local search method without heuristic, and with the heuristic.

Figure 3.7: Graphs illustrating the performance of local search methods when applied to TSPLIB.

Difference in Global Bests			
Local Search Method	**Hypothesis Tested**	**P-value**	**95% Significant?**
2-Opt LS	-H = +H	0.61	No
2h-Opt LS	-H = +H	0.75	No
3-Opt LS	-H = +H	0.53	No
Difference in number of Optimums achieved			
2-Opt LS	-H = +H	0.92	No
2h-Opt LS	-H = +H	0.57	No
3-Opt LS	-H = +H	0.58	No

Table 3.2: Table showing significant differences between local search methods with the heuristic (+H) and without the heuristic (-H) . (Paired Mann-Whitney U-Test, one-tailed to 95% significance, p-value truncated to two decimal places, Bonferroni corrected α is $\frac{0.05}{3} = 0.017$)

To test whether there was a difference between the two situations a Mann-Whitney U-test was performed (see Section 3.4). The results are given in Table 3.2. There were no significant differences between the medians of the two sets and no significant differences between the numbers of optima achieved from the 10 runs. From this it can be inferred that, although some better solutions may have been created, the distribution of the quality of solutions created was the same.

Overall from this small section a number of things can be concluded:

- 3-Opt is the best local search method with or without the heuristic.
- The heuristic does not significantly influence the distribution of the quality of solutions created.
- Above 195 cities all three methods require some assistance in achieving the optimum, either more iterations or additional support.

The TSP has been used for testing many of the Ant algorithms, as discussed earlier in Chapter 2. A short summary of this work is given in Table 3.3. From this table it can be seen that most of the work has been applied to the Symmetric and Asymmetric TSP, both of these are static versions of the problem. However, new versions, such as the dynamic and probabilistic versions, allow new research to be begun on an otherwise

Variant	ACO Algorithm	Year	Authors	Cross-Ref	Main Citation
S	Ant-Density	1991	Dorigo, Maniezzo, Colorni	2.2.1	[Colorni et al., 1991]
S	Ant-Quantity	1991	Dorigo, Maniezzo, Colorni	2.2.1	[Colorni et al., 1991]
S	Ant-Cycle	1991	Dorigo, Maniezzo, Colorni	2.2.1	[Colorni et al., 1991]
S,A	Ant-Q	1995	Gambardella, Dorigo Dorigo	2.2.2	[Gambardella and Dorigo, 1995]
S,A	ACS	1996,97	Gambardella, Dorigo	2.2.3	[Gambardella and Dorigo, 1996]
S,A	AS	1996	Dorigo, Maniezzo, Colorni	2.2.1	[Dorigo et al., 1996]
S,A	MMAS	1997	Stützle, Hoos	2.2.6	[Stützle and Hoos, 1997]
U	AS_{rank}	1997	Bullnheimer, Hartl, Strauss	2.2.4	[Bullnheimer et al., 1997c]
ID	ACO	2001	Guntsch, Middendorf, Schmeck	2.2.8	[Guntsch et al., 2001]
P	ACS	2002	Bianchi, Gambardella, Dorigo	2.2.3	[Bianchi et al., 2002a]
TJ	AS	2002	Eyckelhof, Snoek	2.2.1	[Eyckelhof and Snoek, 2002]
S,A	P-ACO	2002	Guntsch, Middendorf	2.2.13	[Guntsch and Middendorf, 2002b]

Table 3.3: Ant Algorithms that have been applied to the TSP : (S=Symmetric, A=Asymmetric, TJ=Traffic Jam, ID=Insert/Delete, P=Probabilistic, U=Unclear)

heavily studied problem.

3.3.2 Quadratic Assignment Problem

The Quadratic Assignment Problem (QAP) is another optimisation problem that has received a great deal of attention from the Ant algorithms community [Gambardella et al., 1999b]. Like the TSP this problem has been shown to be NP-Complete, and also, not only a hard problem, but among the ”hardest of the hard” [Sahni and Gonzalez, 1976]. The problem was originally introduced in 1957 by Tjalling C. Koopmans and Martin Beckman [Koopmans and Beckman, 1957] as a response to a facilities location problem. Since then it has found a number of other practical applications, for example back-board wiring [Steinberg, 1961], hospital layout [Elshafei, 1977] and scheduling [Geoffrion and Graves, 1976; Carlson and Nemhauser, 1966].

The problem requires the assignment of a set of facilities to a set of locations with given

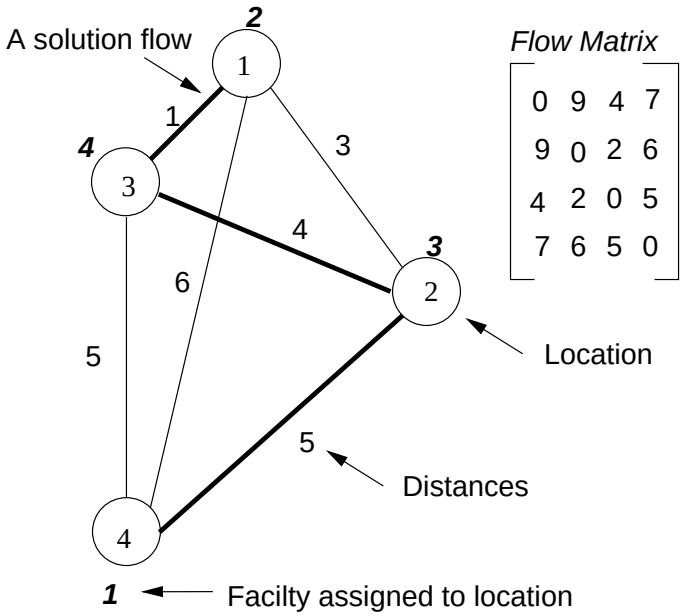


Figure 3.8: An example of a QAP problem of size 5.

distances between the locations and given flows between the facilities. The objective of this scenario is then to allocate the facilities to locations in such a way as to minimise the product between flows and distances. An illustration is given in Figure 3.8.

The QAP can be formally defined as in Definition 3.3.6 [Gambardella et al., 1999b].

Definition 3.3.6 *Given n facilities and n locations, two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{rs}]$, where a_{ij} is the distance between locations i and j and b_{rs} is the flow between facilities r and s, the QAP can be stated as follows:*

$$\min_{p \in \pi} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} a_{p(i)p(j)} \tag{3.4}$$

where π is the set of all permutations (corresponding to the assignments) of the set of integers $\{1, \ldots, n\}$, and $p(i)$ gives the location of facility i in the current solution p. The product $b_{ij} a_{p(i)p(j)}$ describes the cost contribution of simultaneously assigning the facility i to location $p(i)$ and facility j to location $p(j)$.

The term *quadratic* stems from the formulation of the QAP as a Binary Integer Program with the quadratic objective function in Equation 3.5. Let x_{ij} be a binary variable which takes value 1 if facility i is assigned to location j and 0 otherwise. The problem can be formulated as ([PARDALOS et al., 1994]):

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{n} \sum_{k=1}^{n} a_{ij} b_{kl} x_{ik} x_{jl} + \sum_{i=1}^{n} c_{ip(i)} \tag{3.5}$$

where $c_{ip(i)}$ is the cost of placing facility i to location $p(i)$, normally this is left out as it is zero or irrelevant to the application.

Two of the heuristic algorithms that were applied to the QAP before Ant algorithms are CRAFT and GRASP. Computerized Relative Allocation of Facilities Technique (CRAFT) is another constructive heuristic algorithm, which was used for the layout of facilities and was first introduced in [Armour and Buffa, 1963].

Greedy Randomized Adaptive Search Procedure (GRASP) [Li et al., 1994b] has been successfully applied to over 88 QAP problems, and found either the best known solution or an improved solution [Pardalos et al., 1994]. GRASP is an iterated randomised sampling technique in which every iteration produces a new approximate solution to the problem. The procedure can be split into two stages, first the approximate solution is generated and then a local search hones this solution to a local optimum. To get many of the optimal results the procedure is placed within a Branch-and-Bound algorithm. More information on this framework is given in Section 2.2.1.

The QAP has been used many times to test new Ant algorithms. In Table 3.4 the main papers are given alongside the various versions of the algorithm were used. This work is predominantly based on the static versions of the QAP and work on a Dynamic QAP is still young.

In previous Ant algorithms the majority do not use a heuristic when choosing which vertex should next be visited. In [Maniezzo and Colorni, 1999a] a matrix called the *coupling matrix* C is defined that is the product of A and B, $C = A \cdot B^T$, and is used as a heuristic in the formula $\frac{1}{c_{ij}}$. Alternatively a greedy construction heuristic can be used. This requires that the partial objective value is calculated for each option and then the vertex with the best objective value is chosen. In this thesis the first option was chosen, thus no heuristic was used with this domain.

As QAP has been used so much with various versions of Branch and Bound, a key research area has been the calculation of efficient tight lower bound algorithms [Pardalos et al., 1994]. To date there are four main categories of lower bounds that can be used. An overview of these is given below, for more information one should refer to the references given with each description.

ACO Algorithm	Year	Authors	Cross-Ref	Citation Citation
AS-QAP	1994/9	Maniezzo, Colorni	2.2.1	[Maniezzo and Colorni, 1999a]
HAS-QAP	1997/9	Gambardella, Taillard, Dorigo	2.2.1	[Gambardella et al., 1997]
FANT-QAP	1997	Taillard, Gambardella	2.2.5	[Taillard and Gambardella, 1997a]
MMAS-QAP	1997	Stützle	2.2.6	[Stützle, 1997]
ANTS-QAP	1998	Maniezzo	2.2.7	[Maniezzo, 1998]
P-ACO	2002	Guntsch, Middensorf	2.2.13	[Guntsch and Middendorf, 2002b]

Table 3.4: Ant Algorithms that have been applied to the QAP

- *Gilmour-Lawler Bound (GLB) and Related Bounds* are calculated by solving the linear assignment problem $\min_{p\in\Pi}\sum_{i=1}^{n} L_{ip(i)}$, where the matrix L is a special matrix calculated from A and B. (See [Lawler, 1963; Gilmore, 1962].)

- *Eigenvalue Based Bounds* are lower bounds based on the eigenvalues of the two matrices A and B. (See [Rendl and Wolkowicz, 1992; Hadley et al., 1992, 1990])

- *Reformation Based Bounds* based on an iterative method have been proposed in a number of papers. However, most do not run very efficiently, $O(kn^5)$ being the best, where k is the number of iterations.

- *Optimal Reduction Based Bounds* are a class of lower bounds based on optimal reduction schemes proposed in [Li et al., 1994a]. The technique relies on partitioning the matrices into two separate matrices that when added equal the original. Different methods of partitioning yield different lower bounds. This is fairly efficiently computed in $O(n^3)$.

Using the latest branch and bound techniques researchers have managed to prove optimality for problems around $n = 30$, such as nug30 in 2000 [Anstreicher et al., 2002] and Tai25a,Kra30a by Peter Hahn in February 2003. The size of these problems demonstrates how hard these problems are. This having been said, many problems do have good approximate solutions and therefore techniques that can improve on these can make an impact here.

In [Pardalos et al., 1994] a local search technique is described similar to Lin-Kernighan algorithm for the Graph Partitioning Problem (GPP), which is similar to the local search for the TSP. This local search can be run in polynomial time.

Figure 3.9: A Local Search Algorithm for the QAP

input : n, $n \times n$ matrix A, B and a permutation p of size n

output: A local optimal permutation p for the QAP

1 (i) Set $p_0 = p$ and calculate its cost $C(p_0)$. Set $i = 0, g_i = 0$, and $G(i) = 0$, where g_i and $G(i)$ are the step gain and the cumulative gain respectively.

2 (ii) $i = 1$. Initially, select a pair of facilities such that, by exchanging their locations, a positive step gain is obtained, i.e. $g_1 = C(p_0) - C(p_1) > 0$. If no such pair exists then go to (vii), otherwise set $G(1) = g_1$.

3 (iii) $i = i + 1$. For each pair of facilities not already selected, evaluate the step gain by exchanging their locations. Then, select the pair with maximum gain $g_i = C(p_{i-1}) - C(p_i)$. If all facilities have been selected then set $i = i - 1$ and go to (v).

4 (iv) Compute the cumulative gain, $G(i) = \sum_{k=1}^{k=i} g_k$. If $G(i) > 0$; then go to (iii).

5 (v) Select k, such that $G(k)$ is maximum for $0 \geq k \geq i$.

6 (vi) If $k > 0$ and set $p_0 = p_k$ and go to (ii).

7 (vii) We have reached a local optimum for the QAP. Set $p = p_0$ and output p and $C(p)$.

The local search in [Stützle and Dorigo, 1999a] is a 2-Opt algorithm whose relationship to the algorithm in [Pardalos et al., 1994] is analogous to the 2-Opt algorithm for TSP to the Lin-Kernighan local search. The algorithm proposed runs in $O(n^3)$ and their results suggest short bursts of Simulated Annealing or Tabu Search run before this local search helps the local search to get out of worse local optima than it might otherwise find. To calculate the difference when swapping two facilities $p(i)$ and $p(j)$ one can use Equation 3.6, which can be calculated in $O(n)$.

$$\begin{aligned}\Delta(p,i,j) = \quad & b_{ii} \cdot (a_{p(j)p(j)} - a_{p(i)p(i)}) + b_{ij} \cdot (a_{p(j)p(i)} - a_{p(i)p(j)}) + \\ & b_{ji} \cdot (a_{p(i)a_{p(j)}} - a_{p(j)a_{p(i)}}) + b_{jj} \cdot (a_{p(i)p(i)} - a_{p(j)p(j)}) + \\ & \textstyle\sum_{r=1, r\neq i,j}^{n} (b_{ri} \cdot (a_{p(r)p(j)} - a_{p(r)p(i)}) + b_{rj} \cdot (a_{p(r)p(i)} - a_{p(r)p(j)}) + \\ & (b_{ir} \cdot (a_{p(j)p(r)} - a_{p(i)p(r)}) + b_{jr} \cdot (a_{p(i)p(r)} - a_{p(j)p(r)})\end{aligned} \tag{3.6}$$

The efficiency of this calculation can be improved by using information from previous iterations [Taillard, 1995]. If p' is a previous computation with two facilities $p'(k)$ and $p'(l)$ to be swapped, where $\{k,l\} \cap \{i,j\} = \emptyset$, then the move can be evaluated in constant time as shown in Equation 3.7.

$$\begin{aligned}\Delta(p',k,l) = \quad & \Delta(p,i,j) \\ & +(b_{ik} - b_{il} + b_{jl} - b_{jk}) \cdot (a_{p(j)p(k)} - a_{p(j)p(l)} + a_{p(i)p(l)} - a_{p(i)p(k)}) \\ & +(b_{ki} - b_{li} + b_{lj} - b_{kj}) \cdot (a_{p(k)p(j)} - a_{p(l)p(j)} + a_{p(l)p(i)} - a_{p(k)p(i)})\end{aligned} \tag{3.7}$$

The instances that will be used as benchmarks come from a library called QAPLIB [Burkard et al., 1996]. According to [Taillard, 1995] these problems can be categorised into four classes.

- *Unstructured, randomly generated instances* (URGI) : These problems can be recognised by the regular expression 'tai.*a'. These instances are the hardest to solve, although most iterative methods can get within 1-2% of the best known solutions relatively fast.

- *Unstructured instances with grid-distances* (UIGD) : These problems have random flows and distances are calculated via a Manhattan function on a grid.

- *Real-life instances* (RLI) : Among these are instances matching the regular expressions 'ste36.*', 'kra30.*' and 'bur26.*'. These all have in common that the flow matrices are sparse, unlike the randomly generated ones and the remaining ones are clearly not uniformly distributed.

- *Real-life like instances* (RLLI) : Instances fitting the regular expression 'tai.*b' are problems that are like the real life ones mentioned above, the problem with those was their limited size therefore these are larger problems but with similarly distributed flow matrices.

To distinguish between these problems in terms of complexity one can use the flow dominance rule that measures the variation in the flow matrix entries, this is Equation 3.8. A full list of the flow and analogous distance dominance values can be found in Appendix A.

	First-Improvement			Best-Improvement		
Error	0%	1-5%	5-20%	0%	1-5%	5-20%
URGI	25	75	0	25	75	0
URGD	43	43	14	45	45	10
RLI	78	16	6	78	16	6
RLLI	27	73	0	36	64	0

Table 3.5: Table showing the results of the local search methods applied to QAPLIB (134 problems).

$$
\begin{aligned}
fd &= 100 \cdot \frac{\sigma}{\mu}, \\
\text{where} \quad \mu &= \frac{1}{n^2} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij}, \\
\sigma &= \sqrt{\frac{1}{n^2-1} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} (b_{ij} - \mu)^2}
\end{aligned}
\tag{3.8}
$$

To find out how useful the local search method was at solving the problems in QAPLIB, both a First-Improvement and Best-Improvement algorithm were run on the data set. Each run lasted for 1000 iterations, each iteration consisting of one solution being constructed and the local search method called. This was then performed ten times per problem and the best solution achieved was recorded.

The results are summarised in Table 3.5. The first observation is that there is little difference in performance between the First- and Best-Improvement algorithms. The second observation is that a high percentage of the real-life problems are solvable just using this coarse search, although the problems in this group are small (on average the problem size is in the range 16-36 with the largest at 128 items).

In Figure 3.10 bar charts illustrate the results for each version of the local search method. Immediately it is obvious that over half of the problems are solvable by this method and over 80% are solved to less than 5% of the known optimum. This illustrates just how strong the local search is for this problem and that using it in a hybrid means it is consistency and speed that are the results of interest rather than quality itself on the majority of these problems.

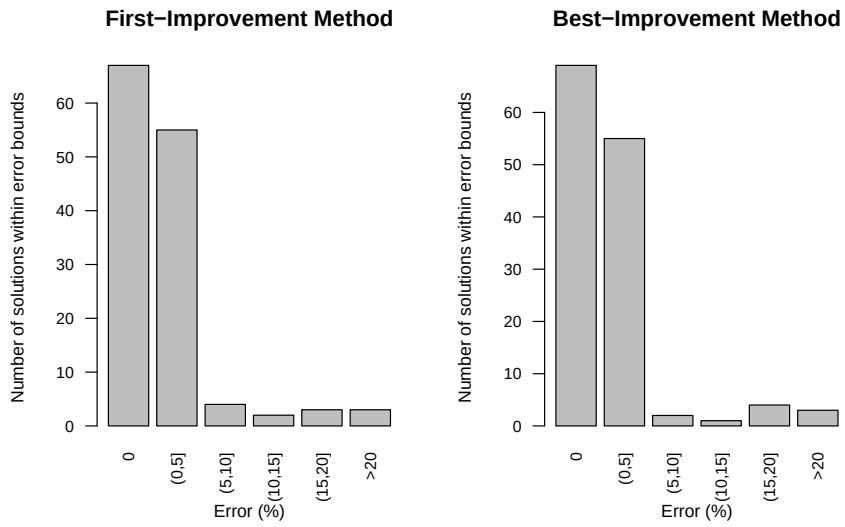


Figure 3.10: Bar charts illustrating the results of applying the local search methods to the QAPLIB. (The notation $(x,y]$ means $x <$ Percentage Error $<= y$)

3.3.3 Talent Scheduling

The definition of the Talent Scheduling (TS) problem evolved from the Orchestra Rehearsal Scheduling Problems (ORSP) that was first defined in [Adelson et al., 1976] as a general formulation for a number of applications, such as archaeology and actor scheduling on film sets. First the definition of this problem is given with an example from [Smith, 2003] in Table 3.6.

Definition 3.3.7 *An $m \times n$ matrix contains elements T_{ij} which are either* 0 *or* 1*. It is required to permute the m rows of the matrix so as to minimise the sum of the differences between the row numbers of the first and last non-zero elements in each column. The zero elements between the first and last non-zero elements in each column are called* hold *elements.*

A more complex form of the problem occurs when the rows of the matrix have weights $d_i, i = 1,\ldots,m$ and it is required to minimise the total over all columns of the inclusive sum of the row weights between the first and last non-zero elements in each column. This is the Orchestra Rehearsal Scheduling Problem (ORSP).

This qualitative definition translates into the following quantitative version.

Definition 3.3.8 *Let $f(p_1, p_2, \ldots, p_i, \ldots, p_m)$ where $p_i = 1$ if piece i has already been played $p_i = 0$ otherwise, be the total man-hours required for the remaining pieces*

Piece	1	2	3	4	5	6	7	*Cost*
Player 1	1	1	0	1	1	1	1	*10*
Player 2	1	0	1	0	0	0	1	*20*
Player 3	1	0	0	0	0	1	0	*5*
Player 4	1	1	1	1	0	0	1	*1*
Player 5	1	0	0	1	1	1	1	*15*
Duration	2	4	1	3	3	2	5	

Table 3.6: An example of a Talent Scheduling Problem (the column in italics could be removed to transform it into a Orchestra Rehearsal Scheduling problem).

using an optimal policy. If the following is defined:

$$a \oplus b = 0 \quad \textit{if } a = b = 0,$$
$$a \oplus b = 1 \quad \textit{otherwise}$$

and define

$$l_j = 1 \quad \textit{if } \textstyle\sum_{i \text{ s.t. } p_i=1} T_{ij} > 0 \textit{ and } \sum_{i \text{ s.t. } p_i=0} T_{ij} > 0,$$
$$l_j = 0 \quad \textit{otherwise}$$

then

$$f(p_1,\ldots,p_m) = \min_{i \text{ s.t. } p_i=0} \left[p_i \left(\textstyle\sum_j (T_{ij} \oplus l_j) \right) + f(p_1,\ldots,p'_i,\ldots,p_m) \right]$$

Note that one needs to add on d_i for each player who is either playing in piece i ($T_{ij} = 1$), or who has played in a piece and has still to play in another piece ($l_j = 1$) or both.

A complete algorithm could start with $f(0,0,\ldots,0) = 0$ and compute the 2^m state values successively ending with $f(1,1,\ldots,1)$, and hence find the optimal schedule although this may take a substantial amount of time.

More realistic methods to solve this problem have taken several forms. It was modelled as a Constraint Satisfaction Problem in [Smith, 2003] and then a planning and checking algorithm was applied in [Gregory et al., 2004] and both proved successful at the problem, although they were both very slow. In the former it should be noted that a priori work was done to make the problem easier, as the size grew, by identifying symmetries and implied constraints.

The ORSP can be stated another way, which is more practical for implementation, as follows:

Definition 3.3.9 *Given n pieces of music and m players, one is given a matrix T that is a boolean matrix where t_{ij} indicates whether the player performs in a particular piece. There is a cost associated with each piece $\delta_i, 1 \leq i \leq n$ that represents the duration of the piece.*

The *Talent Scheduling Problem* [Cheng et al., 1993] is identical except for additional cost for waiting per player $c_i, 1 \leq i \leq n$. In notation pieces and players are renamed shooting days and actors accordingly. In this paper it was shown to be Strongly NP-Hard, like the QAP, by a reduction to the Optimal Linear Arrangement Problem (OLAP). A pertinent observation is that the ORSP is simply the Talent Scheduling problem with a uniform cost vector, and therefore the latter is a generalisation of the ORSP. For simplicity it will be assumed this vector is the unit vector, but this is an arbitrary choice.

These problems also fall within another field of problems called Pattern Sequencing Problems (PSP). In [Fink and Voß, 1999] the ORSP was reduced to the Pattern Sequencing subproblem called Simultaneously Open Stacks Problem (PSP-SOS), and the Talent Scheduling Problem was reduced to a generalisation of the Average Order Spread (PSP-AOS) problem. A consequence of this work was that the ORSP was not shown to be NP-Hard.

In [Cheng et al., 1993] a number of useful heuristics, bounds and a local search were described. For the rest of this section pieces and players will be referred to as actors and days.

The bound is fairly complicated and relies on a number of observations. The bound assumes a construction algorithm moving down a search tree. A path with less than n values is known as a *partial solution*. Let J_i be the shooting day which is scheduled to be filmed in the i-th day of the shoot. Each node in the graph is defined as a partially defined solution and at the root of the graph are schedules with only J_1, the second level has nodes with J_1 and J_n determined, third level has J_2 and J_{n-1} and so on, working outside in towards a completed schedule.

The first observation splits a partial solution P into two subsets, *Late* in Equation 3.9 and *Early* in Equation 3.10, where $\lceil j \rceil$ represents the smallest integer greater than or equal to j. J_k is the set of possible days to add to the partial solution in the k-th step.

$$E(P) = \{k \in \{1,\ldots,n\} | J_k \in P \text{ and } k \leq \left\lceil \frac{n}{2} \right\rceil\} \tag{3.9}$$

$$L(P) = \{k \in \{1,\ldots,n\} | J_k \in P \text{ and } k > \left\lceil \frac{n}{2} \right\rceil\} \tag{3.10}$$

Then if an actor is required in both an early day and a late day, both of which have already been scheduled then the maximum number of hold days for that actor is determined.

Let P be any partial schedule and define

$$\begin{aligned} \varepsilon_i &= \begin{cases} 1, & \text{if actor } i \text{ is required in a day in E(P)} \\ 0, & \text{otherwise} \end{cases} \\ \lambda_i(P) &= \begin{cases} 1, & \text{actor } i \text{ is required in a day in L(P)} \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3.11}$$

Also, define

$$\begin{aligned} e_i(P) &= \begin{cases} \text{first day in P where actor } i \text{ is required}, & \text{if } \varepsilon_i(P) = 1, \\ 0, & \text{otherwise}, \end{cases} \\ l_i(P) &= \begin{cases} \text{last day in P where actor } i \text{ is required}, & \text{if } \lambda_i(P) = 1, \\ 0, & \text{otherwise}. \end{cases} \end{aligned} \tag{3.12}$$

Finally, if we define

$$D_i^e(P) = \text{number of days } j \in E(P) \text{ which are hold days}, \tag{3.13}$$

and similarly

$$D_i^l(P) = \text{number of days } j \in L(P) \text{which are hold days}, \tag{3.14}$$

we can then define a lower bound for the cost due to hold days of a completed schedule S from a partial solution P:

$$\begin{aligned} LB(S) \geq \sum_{i=1}^{m} c_i \{ \quad & \varepsilon_i(P)\lambda_i(P) \left[l_i(P) - e_i(P) + 1 - \sum_{j=1}^{n} t_{ij} \right] + \\ & [1 - \varepsilon_i(P)\lambda_i(P)] \left[D_i^e(P) + D_i^l(P) \right] \} \end{aligned} \tag{3.15}$$

Without Heuristic							
Instance ($m \times n$)	**Min.**	**1st Q.**	**Median**	**Mean**	**St. Dev.**	**3rd Q.**	**Max.**
ORSP (5*x*9)	0.0	17.7	17.7	22.8	8.7	29.4	35.3
	(0.00)	(0.05)	(0.09)	(0.09)	(0.06)	(0.14)	(0.18)
TS 1 (10*x*13)	75.9	94.3	104.6	106.5	16.3	113.8	142.5
	(0.02)	(0.25)	(0.58)	(0.62)	(0.43)	(1.04)	(1.27)
TS 2 (8*x*20)	182.9	210.3	223.3	224.4	18.8	237.7	261.0
	(0.01)	(0.90)	(1.21)	(1.22)	(0.61)	(1.77)	(2.00)
With Heuristic							
ORSP (5*x*9)	0.0	11.8	17.7	17.7	7.96	23.5	29.4
	(0.04)	(0.14)	(0.18)	(0.19)	(0.11)	(0.23)	(0.47)
TS 1 (10*x*13)	1.1	13.8	19.5	21.0	9.89	27.6	42.5
	(0.08)	(0.55)	(1.82)	(1.93)	(1.48)	(3.39)	(4.13)
TS 2 (8*x*20)	54.8	76.0	89.0	85.3	14.3	95.9	105.5
	(0.05)	(1.12)	(5.39)	(4.20)	(2.93)	(6.50)	(8.34)

Table 3.7: Results for generating random solutions with and without the Hamming Distance heuristic. (Q=Quartile. The percentage over optimum is given to one decimal place, and the time taken to find best solution in brackets to two decimal places.)

A branch-and-bound solver was constructed to test this bound. It was discovered that this bound alone did not sufficiently restrict the search space and for problems of only 10 shooting days it took a significant amount of time to complete.

The first heuristic is simple, pick the shooting day probabilistically in inverse proportion to its increase to the lower bound, which would be calculated dynamically. A less computational intensive heuristic is to choose the next shooting day with the smallest Hamming Distance from the last day scheduled, called the *Hamming Distance Heuristic*. This heuristic also has the advantage of greater local awareness scoring potential moves more finely than the lower bound. The quality of this heuristic is illustrated in Table 3.7, where results are shown for the original problems that were used in [Smith, 2003] and [Gregory et al., 2004].

The experiment was carried out by performing 25 runs for each problem and letting each run last for 1000 iterations. The first batch generated solutions using a uniform random distribution and the second batch used a distribution proportionally biased by the Hamming-Distance heuristic. The significance tests were performed using a Mann-Whitney U-Test due to the small number of runs and the distribution of results did not fit a Normal distribution (for more information see Section 3.4).

Instance	Hypothesis	p-value	95% Significant?
ORSP (5x9)	+H $<$ -H	0.0070	Yes
TS 1 (10x13)	+H $<$ -H	< 0.0001	Yes
TS 2 (8x20)	+H $<$ -H	< 0.0001	Yes

Table 3.8: Table showing the significance between the medians of using the heuristic as opposed to not. (P-values are given to four decimal places. +H means a heuristic was used and -H means no heuristic was used. Bonferonni corrected α is $\frac{0.05}{3} = 0.017$.)

From Table 3.8 one can see that the heuristic does make a significant difference. The increase in time to generate the solutions is the extra calculation time required to evaluate the possible vertices at each step in the construction of a solution. For the larger problems the use of the heuristic makes a significant difference, increasing the mean quality by over 50%.

In [Cheng et al., 1993] a pairwise local search is described that compares the $\frac{n(n-1)}{2}$ neighbours in the neighbourhood of swapping the positions of two days, and then moving strictly down towards a local optimum. Unfortunately there are no details to create an efficient implementation, therefore Chapter 4 will introduce one.

To complete the empirical investigations later in the thesis it will be necessary to generate a collection of random Talent Scheduling problems. Figure 3.11 shows the algorithm that was used to generate random problems for the Talent Scheduling and Orchestra Rehearsal Problems. This algorithm requires the user to specify a range of values to define the duration and cost vectors. The Orchestra Rehearsal Problem can be formulated as a TS problem with a unit cost vector. Therefore the same algorithm can cope with both problems, either explicitly as written in the pseudo-code algorithm, or implicitly, by inputting the cost range as a parameter of $(c_{min}, c_{max}) = (1,1)$. The algorithm makes use of a function *random(a,b)* that generates a random number from a uniform distribution in the range (a,b).

3.4 Data Analysis

Two groups of data will be collected, *ratio* and *interval*. Ratio data is any data that has a specific zero and a common example of this is any kind of time measurement, such as CPU Time. Interval data, on the other hand, is measured on a scale with a

Figure 3.11: Pseudo-code for a random Talent Scheduling Problem generator

input : t : problem type from the set {TS, ORSP}
input : n : number of shooting days, m : number of players
input : $\delta_{min}, \delta_{max}$: duration range, c_{min}, c_{max} : cost range
output: Three completed matrices: occurrence matrix T, durations Δ and costs C

```
for i = 1,...,n do
    for j = 1,...,m do
        if random(0,1) > 0.5 then
            T(i,j) = 1;
        else
            T(i,j) = 0;
        endif
    endfor
endfor
for i = 1,...,n do
    Δ(i) ← random(δ_min, δ_max);
endfor
if t =TS then
    for i = 1,...,m do
        C(i) ← random(c_min, c_max);
    endfor
else
    C ← 1;
endif
```

defined, but arbitrary, zero. One example of interval data is the quality of the best solution produced by a particular run, because the objective function is defined by the researcher and is limited to the distribution of the set of objective values.

When a set of runs has been executed and the data extracted it is then necessary to decide what distribution the data fits. If the data fits a Normal Distribution then *parametric* statistical tests can be used upon the data, otherwise *non-parametric* tests must be used.

Statistics are functions on samples while parameters are functions on populations. Therefore one can estimate parameters using statistics. There are two reasons for performing empirical experiments, to test a hypothesis or to estimate a parameter. The following definitions and couple of paragraphs have been paraphrased from [Cohen, 1995].

Hypothesis Testing Answer a yes-or-no question about a population and assess the probability that the answer is wrong.

Parameter Estimation Estimate the true value of a parameter given a statistic.

In this thesis Hypothesis Testing is the primary reason for using statistical tests. In Hypothesis Testing, two statements are given. The first, called the *Null Hypothesis* H_0, states that there will be no difference in the two distributions being tested. The second, called the *Alternative Hypothesis* H_1, is what is being tested and may be one of three conditions: given two distributions, X and Y, a *two-tailed* test asks is X equal to Y; a *one-tailed* test asks either is X greater than Y or is X less than Y.

To test the claim of the Alternative Hypothesis a sample of a given size must be taken. The sample size N will vary between experiments depending on the statistical test and degree of confidence that is required in the result. As a general rule the greater the value of N the better, and values between 20 and 30 for each sample can be regarded as reasonable sizes. In all of the experiments in later chapters 25 samples are taken per result, this balances the need for significance with the amount of time required to collect the samples.

Where necessary a summary of the data is given that indicates both the range and the diversity of the data. An example is given in Table 3.9. The tables will consist of six statistics: minimum, 1st quartile , mean (μ), standard deviation (σ), median, 3rd quartile and maximum. The median is the second quartile and the three quartiles are

Example Data Set						
Min.	1st Quartile	Median	Mean	St. Dev	3rd Quartile	Max.
2.0	220.2	426.5	463.1	287.2	704.0	1000.0

Table 3.9: Example of a summary table for a distribution.

computed by ranking all the data and then taking the $\frac{N}{4}$-th, $\frac{N}{2}$-th and the $\frac{3N}{4}$-th data item respectively. Given a data set $x = x_1, \ldots, x_n$, the mean is calculated by $\frac{\sum_{i=1}^{N} x_i}{N}$, where x_i is the i-th item. The related standard deviation is calculated using the following formula $\sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})}{N-1}}$, where $\bar{x}$ is the mean of the sample. From this set of values a number of useful descriptive ranges can be found.

- The *range*, $maximum - minimum$, gives the extreme values of the distribution.
- The *inter-quartile range*, $3rdquartile - 1stquartile$, gives the range of values in which the bulk of the results for non-parametric data lies.

Accompanying the summary table will be visualisations of the data. A very useful graph in this instance is the box-and-whisker plot. This type of graph shows the median, inter-quartile range, bounds those points within $\frac{3}{2}$ times the inter-quartile range and finally displays dots for those points considered outliers. An example of this type of plot is given in Figure 3.12. For scatterplots, the points, unless otherwise specified, will be the medians of the distributions, while the error bars represent the inter-quartile range. The reason for this is that the data does not fit a Normal distribution and therefore to draw the mean and standard deviations could be misleading.

If the data is thought to be parametric in nature then it must fulfil three criteria:

- The level of measurement must be at least interval.

  This criterion is fulfilled because the data is either interval or ratio.

- The sample data are drawn from a population with a Normal Distribution.

  For small distributions (less than approximately 25 samples) if the distribution is thought to be Normal, the t-test can be used to correct minor deviations from this assumption. Alternatively, the Central Limit Theorem could be used to enable this assumption to be made but this would require significantly more sampling which would take an unacceptable amount of time.

- The variances of the two samples are not significantly different (*Homogeneity of*

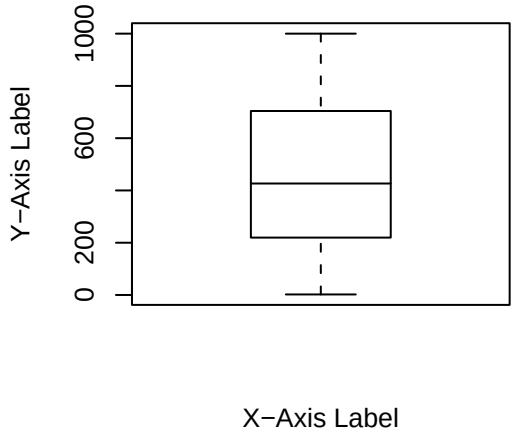


Figure 3.12: Example of a box-and-whisker plot for visualising the distribution of a sample.

Variance)

This is most important if the two samples are not of the same size. All samples taken for a particular experiment in this thesis will be of the same size.

The advantage of being able to use parametric data is that it enables one to use statistical tests such as the t-test that allow greater confidence in any conclusions drawn about the population. However if the samples are large, non-parametric tests can be as good.

In Chapter 4 parametric tests will be used as the data was easily collected in sufficient quantity to be able to fulfil the parametric assumptions stated above. In later chapters it was found that the median was a better descriptor of the data than the mean and that the data was not able to be gathered in sufficient quantity to be able to fulfil the parametric assumptions, thus non-parametric tests were used.

It is important at this point to mention that all the data gathered is from an *unrelated* design. This means that each experiment is independent of the others. The reason for this is that different random numbers are used to seed the experiments as many experiments are run in parallel. There are no significant disadvantages of this method but it does alter the set of statistical tests available for use with the collected data.

If the data being dealt with is parametric the Unrelated t-test will be used as the statistical test. The statistical test equations and conditions come from [Coolican, 1994].

To use the t-test for unrelated data there are some conditions that need to be fulfilled.

- Level of data must be ratio or interval.
- Design of experiment must be unrelated.
- Data must fulfil the parametric assumptions.

Having fulfilled these conditions the justification for using the t-test would be that the data fits the Normal distribution and that the variances of the two sets are not too different. The second condition can be dropped if the number of trials is the same or very close. Equation 3.16 gives the Unrelated t-test for two samples A and B, of size N_A and N_B respectively. $\bar{x}_A$ and $\bar{x}_B$ are the means of the two distributions.

$$\text{Unrelated t} = \frac{|\bar{x}_A - \bar{x}_B|}{\sqrt{\left[\frac{\left(\Sigma x_A^2 - \frac{(\Sigma x_A)^2}{N_A}\right) + \left(\Sigma x_B^2 - \frac{(\Sigma x_B)^2}{N_B}\right)}{(N_A + N_B - 2)}\right]\left[\frac{N_A + N_B}{(N_A)(N_B)}\right]}} \tag{3.16}$$

If these conditions are not met the Mann-Whitney U-test will be used. This compares the medians of the two distributions in a non-parametric way. This test comes with certain requirements which must be met by the data.

- Type of data must be ordinal and must be meaningfully rankable.
- The type of experiment must use unrelated samples.
- The size of the sample should be greater than or equal to 20 samples.

The Mann-Whitney U-test deals with the ranks of particular data items. Therefore if the data contains large numbers of equal data items then the test is less reliable. To overcome this the samples should meet the size requirements. The recipe below shows how the test is used on two samples A and B. The resulting z-score can be compared with the Normal Distribution for critical values.

1. Rank all scores as one group.
2. Find the sum of ranks in group A (R_A) and group B (R_B).
3. Calculate $U_A = N_A N_B + \frac{N_A(N_A+1)}{2} - R_A$.
4. Calculate $U_B = N_A N_B + \frac{N_B(N_B+1)}{2} - R_B$.
5. Select the smaller of U_A and U_B and call it U.

6. Calculate $N = N_A + N_B$.
7. Calculate $z = \frac{U - \frac{N_A N_B}{2}}{\sqrt{\left(\left[\frac{N_A N_B}{N(N-1)}\right] \cdot \left[\frac{N^3 - N}{12} - \sum T\right]\right)}}$, where $T = \frac{t^3 - t}{12}$ each time a number of values are tied at a particular rank and t is the number of times the value occurs.

Besides testing distributions it will also be necessary to test correlations. By testing the correlation between two sets of results it is hoped to discover whether two variables X and Y are related. This does not test whether X is better than Y or visa-versa, but whether, for a given change in some variable, X behaves in a similar way to Y. Correlations should never be calculated alone and are susceptible to small N, where N is the number of samples. To calculate correlations the Pearson Product-Moment Correlation Coefficient was used. The coefficient r is calculated as shown in Equation 3.17, where X and Y are two sets of interval or ratio data, which are in related pairs.

$$r = \frac{N\sum(XY) - \sum X \sum Y}{\sqrt{[N\sum X^2 - (\sum X)^2][N\sum Y^2 - (\sum Y)^2]}} \tag{3.17}$$

The Pearson Product-Moment Correlation was chosen instead of a rank correlation method because the correlations should be similar in scale, as well as in direction. Therefore it was more appropriate to use a non-rank correlation method. Hypothesis testing can also be performed for this correlation method. For this the Fisher Test is used which converts the product-moment coefficient value to a probability for use with the Z test.

All the statistical tests shown will return p-values. A *p-value* represents the residual uncertainty in the conclusion drawn, meaning the chance that the Null Hypothesis was discarded when it was in fact true. The smaller the p-value the lower the uncertainty in the conclusion. Normally one regards a p-value less than 0.05 as conclusive evidence that the Null Hypothesis is correctly discarded, this is called a *statistically significant* result. When choosing the size of sample to take from the population it is important to note that in Hypothesis Testing, once there is enough evidence to decide a certain conclusion to within 95% confidence, the only reason to take a larger sample would be to increase this confidence; the benfit of which is normally a small gain in confidence, at the expense of a great deal of extra time performing more experiments. Therefore the sample sizes chosen for the experiments in this thesis will reflect this issue.

Unsurprisingly, as decisions are being made, errors can creep into the results. As Hypothesis Testing is binary there are two sorts of errors, Type I and Type II. Type I

Sample A	Sample B
13.4	13.1
13.6	13.1
13.2	13.1
13.7	13.6
13.7	13.4

Table 3.10: Example data of two unrelated samples. [Coolican, 1994]

errors, also referred to as α are made when the Null Hypothesis is rejected when it is true (Equation 3.18).

$$\alpha = Pr(\text{Reject } H_0 | H_0 \text{ is true}) \tag{3.18}$$

Type II errors are when the Null Hypothesis is not rejected but it is false, these are also known as β errors (Equation 3.19).

$$\beta = Pr(\text{Fail to reject } H_0 | H_0 \text{ is false}) \tag{3.19}$$

In general it is desirable to protect α by requiring a confidence level of 95%, infering an $\alpha = 0.05$. By setting the confidence threshold too high the probability of rejecting H_0 becomes too great for the results to be useful. If the confidence threshold is too low then the probability of a Type II errors increases. Therefore a trade-off is made between these two errors and normally a confidence level of 95% is acceptable.

Parametric tests such as the t-test generally have more *power* than non-parametric tests because they deal with the actual figures and not just the ranks of the individual data items. An example of this is given below using the data in Table 3.10. If one was to perform a t-test on the data there would not be a significant difference at the 10% confidence level (p-value=0.1034, which is just greater than 0.10), but the Mann-Whitney U-test would be significant (p-value=0.0877, which is less than 0.10). Therefore if we assume that the Null Hypothesis was correct and that there was no difference between the two sets, the non-parametric test has exhibited a Type I error. One could remove this error by increasing the sample size.

For both Hypothesis Testing of data and correlation co-efficients the Bonferroni correction is applied. The Bonferroni correction is a multiple-comparison correction used

when several dependent or independent statistical tests are performed simultaneously. When doing such multiple comparisons it is important to protect against spurious positives and to take into account the number of comparisons being performed. For example, given 18 tests of the hypothesis there is a 60% probability of finding one or more significant results. However, the drawback is that significant results can be thrown away. For example, at the 95% confidence level 18 problems may be found to be significant but with a p-value of 0.01, according to Bonferroni none of these results would be significant. To make the trade-off between these Type I and Type II errors, the Bonferroni corrected results will represent a worst case scenario.

In all tables where multiple comparisons are made the Bonferroni corrected alpha value will be stated and a small 'B' will appear superscript to the p-value to indicate that the value is no longer significant when the correction is made. Given n dependent or independent experiments for a particular hypothesis, the Bonferroni correction adjusts α, in this case 0.05, by dividing through by n. This is shown in Equation 3.20.

$$\alpha_{bonferroni} = \frac{\alpha}{n} \tag{3.20}$$

To make the statistical calculations more efficient the statistical package R [Gentleman and Ihaka, 1997] will be used.

3.5 Summary

In this chapter the experimental methodology for this thesis has been described with a discussion of the major issues. A number of experimental choices have been explained including the choice of data sets and the design choices associated with the implemented test-suite. These choices will be form the basis for Chapters 4 onwards. The chapter concluded with a description of the basic statistics that will be used to give evidence for the significance of the work pursued later in this thesis.

Chapter 4

A Local Search Method for Talent Scheduling

This chapter introduces an efficient local search method for the Talent Scheduling Problem. Although there had been one attempt alluded to in [Cheng et al., 1993], there was not sufficient information from this paper to create an implementation. Therefore this work was undertaken to create an efficient local search method that did not require complete recalculation of the objective value of the solution. This chapter also explores the benefits of using both a First-Improvement and a Best-Improvement method for solving Talent Scheduling problems.

In Section 4.1 the main background of the problem is revisited, followed in Section 4.2 by the local search method itself. Section 4.3 describes four versions of this method to be compared and afterwards the results are discussed. In Section 4.4 the main conclusions of the study are presented. Finally, Section 4.5 provides a summary of the chapter.

4.1 Introduction

In Section 3.3.3 it was stated that a pairwise interchange local search method was alluded to in [Cheng et al., 1993]. Unfortunately details of the method were not given so the objective of this chapter is to construct a viable local search method for the Orchestra Rehearsal Scheduling Problem (ORSP) and Talent Scheduling (TS). This local search method will be combined with the Ant algorithms and used in later chapters.

Shooting Day	1	2	3	4	5	6	7	*Cost (C)*
Actor 1	1	1	0	1	1	1	1	*10*
Actor 2	1	0	1	0	0	0	1	*20*
Actor 3	1	0	0	0	0	1	0	*5*
Actor 4	1	1	1	1	0	0	1	*1*
Actor 5	1	0	0	1	1	1	1	*15*
Duration (δ)	2	4	1	3	3	2	5	

Table 4.1: An example of an Talent Scheduling Problem. (The column in italics could be removed to transform it into a Orchestra Rehearsal Scheduling problem.)

For the purposes of this chapter the Talent Scheduling nomenclature from the Section 3.3.3 will be used.

To begin with the initial definition is given below as a reminder of the Talent Scheduling problem.

Definition 4.1.1 *Given n shooting days and m actors, one is given a matrix T that is a boolean matrix where t_{ij} indicates whether the actor i performs on a particular shooting day j. There is a [duration] cost associated with each shooting day $\delta_i, 1 \leq i \leq n$ that represents the duration of the shooting day. A vector C is defined where $c_i, 1 \leq i \leq n$ is a cost of actor i per shooting day. The problem is to minimise the number of shooting days that the actor is not performing on, after they have turned up for their first shooting day in the schedule. This is the* Talent Scheduling Problem. *[Cheng et al., 1993]*

The Orchestra Rehearsal Scheduling Problem is similar except there are is no cost vector. This can be considered a subclass of the TS problem by making the cost vector uniform. An example of such problems can be seen in Table 4.1.

4.2 The Local Search

The method consists of two parts. Subsection 4.2.1 demonstrates how to create a function that will incrementally calculate the gain for each move made in the local search. The second part then combines this function with both a First-Improvement and a Best-Improvement local search method, discussed in Subsection 4.2.2.

4.2.1 Incremental Gain Function

Let the function that calculates the exchange of two shooting days in a solution be $Gain = \Delta(\pi, i, j)$, taking a solution π and two pointers i and j. If $i > j$ then the pointers are switched. The function consists of, for each actor a in turn, calculating the gain (g) incurred by exchanging two shooting days (see Equation 4.1).

$$Gain = \Delta(\pi, i, j) = \sum_{a=0}^{m} g(a, \pi, i, j) \tag{4.1}$$

The first stage is to find the previous and following shooting days where the actor is performing either side of the positions i and j. The function $l(\cdot)$ returns the previous shooting day and $n(\cdot)$ returns the next shooting day that the actor is performing. This creates the four variables defined below.

- $l(a,i)$,$l(a,j)$: point to the last shooting day the actor a performed before i and j. The value returned by $l(a,i)$ should be in the range of $[0,i)$, and $l(a,j)$ should be in the range (i,j). If the actor has yet to be in a shooting day before position i or j then a value of -1 is returned.
- $n(a,i)$,$n(a,j)$: point to the next shooting days when the actor a performed. For $n(a,i)$ the value returned should be in the range (i,j) and for $n(a,j)$ the value should be in the range $(j,n]$. If the actor does not perform after position i or j then a value of -1 is returned by the function.

Including the two pointers passed through in the function definition this results in the following six variables (illustrated in Figure 4.1):

- $\pi_i(a)$,$\pi_j(a)$: pointers to shooting days at positions i and j ($i < j$) for actor a.
- $\hat{l}(a,i)$,$\hat{l}(a,j)$: decision variables that return TRUE if $l(a,i)$, $l(a,j)$ point to a valid shooting day in the solution π for actor a, FALSE otherwise.
- $\hat{n}(a,i)$,$\hat{n}(a,j)$: decision variables that return TRUE if $n(a,i)$, $n(a,j)$ point to a valid shooting day in the solution π for actor a, FALSE otherwise.

From the $2^6 = 64$ possible combinations of $\{\hat{l}(a,i), \pi_i(a), \hat{n}(a,i), \hat{l}(a,j), \pi_j(a), \hat{n}(a,j)\}$ only twelve result in changes to the objective value of the solution. The others result in no change to the objective value or are not permitted by the logic of the problem. For each of these twelve combinations a particular gain calculation is performed that takes

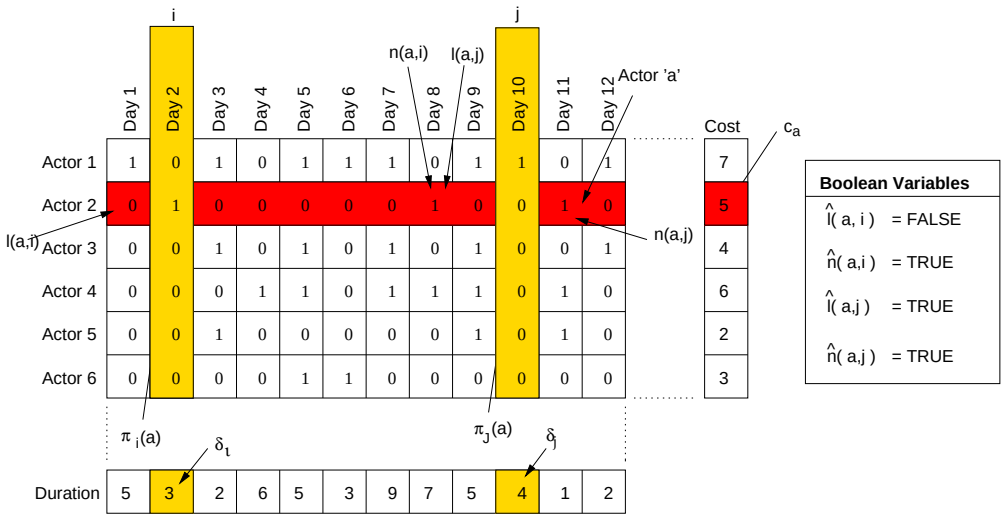


Figure 4.1: Figure illustrating the assignment of six variables, for the purposes of calculating the gain after exchanging two shooting days in a schedule.

into account the shooting days affected by the exchange, all of which are displayed in Table 4.2.

From Table 4.2 it is clear that all instances requiring a calculation have in common that $\hat{n}(a,i) = \hat{l}(a,j)$. As such it is possible to combine these into a single variable. However, for the purposes of explaining the calculations it was decided to leave these as separate entities. With regards to an implementation only one variable is required.

An example is now given using the information from Figure 4.1, where actor number 2 is selected ($a = 2$). The highlighted row gives the following variable values: $l(a,i) = -1$, $\hat{l}(a,i) = FALSE$, $\pi_i(a) = 1$, $n(a,i) = 8$, $\hat{n}(a,i) = TRUE$, $l(a,j) = 8$, $\hat{l}(a,j) = TRUE$, $\pi_j(a) = 0$, $n(a,j) = 9$, $\hat{n}(a,j) = TRUE$.

Therefore the variable set for actor number 2 is $\{0,1,1,1,0,1\}$ which corresponds to line 7 of Table 4.2. The cost for Actor 2 on days 2 and 10 is c_a, which is equal to 5, and the day durations on $i = 2$ and $j = 10$ are $\delta_i = 3$ and $\delta_j = 4$. Therefore using the selected equation gives the calculation in Equation 4.2.

$$g(2) = -1 * (c_a \cdot \sum_{k=3}^{7} \delta_k + 4 \cdot 5) \tag{4.2}$$

Expanding out the summation gives:

$\hat{l}(a,i)$	$\pi_i(a)$	$\hat{n}(a,i)$	$\hat{l}(a,j)$	$\pi_j(a)$	$\hat{n}(a,j)$	**Gain Per Actor** $g(a)$
0	0	0	0	1	1	$c_a \cdot \sum_{k=i}^{j-1} \delta_k$
0	0	1	1	0	1	$c_a \cdot (\delta_i - \delta_j)$
0	0	1	1	1	0	$c_a \cdot (\sum_{k=i+1}^{n(a,i)-1} \delta_k - \sum_{k=l(a,i)+1}^{j} \delta_k)$
0	0	1	1	1	1	$c_a \cdot \sum_{k=i}^{n(a,i)-1} \delta_k$
0	1	0	0	0	1	$-1 * (c_a \cdot \sum_{k=i+1}^{j} \delta_k)$
0	1	1	1	0	0	$c_a \cdot (\sum_{k=l(a,j)+1}^{j-1} \delta_k - \sum_{k=i+1}^{n(a,i)-1} \delta_k)$
0	1	1	1	0	1	$-1 * (c_a \cdot \sum_{k=i+1}^{n(a,i)-1} \delta_k + \delta_j c_a)$
1	0	0	0	1	0	$-1 * c_a \cdot \sum_{k=i}^{j-1} \delta_k$
1	0	1	1	0	0	$c_a \cdot (\delta_j - \delta_i)$
1	0	1	1	1	0	$-1 * (c_a \cdot \sum_{k=l(a,j)+1}^{j-1} \delta_k + \delta_i c_a)$
1	1	0	0	0	0	$c_a \cdot \sum_{k=i+1}^{j} \delta_k$
1	1	1	1	0	0	$c_a \cdot \sum_{k=l(a,j)+1}^{j-1} \delta_k + \delta_j c_a$

Table 4.2: Calculations for each combination of $\{\hat{l}(a,i),\ \pi_i(a),\ \hat{n}(a,i),\ \hat{l}(a,j),\ \pi_j(a),\ \hat{n}(a,j)\}$. The highlighted row is used in the example calculation.

$$g(2) = -1 * (5 \cdot (2+6+5+3+9) + 4 \cdot 5) \tag{4.3}$$

Giving a gain of -145. Therefore swapping day 2 and day 10 for actor number 2 would benefit the solution by 145 units. These calculations would be performed for each actor to get the total gain for the swap.

The full procedure can be seen in Procedure `Calculate_Gain` and will be referred to as the *Incremental Gain Calculation* (IG) for the rest of this chapter. In this procedure the functions *findPreviousPiecePerformed(n)* and *findNextPiecePerformed(n)* calculate the $l()$ and $n()$ functions respectively. This procedure is then plugged into the same harness as the QAP local search procedure allowing the same parameters to be used. As a consequence two versions of the local search method can be investigated, the First-Improvement (FI) version in Figure 4.4, and the Best-Improvement (BI) version in Figure 4.5.

4.2.2 The Local Search Methods

The Best-Improvement method searches the entire neighbourhood for the exchange that maximises the gain. The advantage of this is that the best possible chance is given to find the best local optimum in the current search neighbourhood. The disadvantage of this method is that it tests all $n-1$ possibilities for each player before making the choice, which as n increases becomes expensive. The alternative is to pick the first exchange that shows a negative gain, a method known as First-Improvement. The advantage of this method is that only in the worst case does the method look at all $n-1$ choices, but this comes at the price of sometimes ending up in local minima which are not optimal.

The Best-Improvement version of the local search method runs in $\theta(n^2)$, where $\theta(\cdot)$ is the tight asymptotic upper bound. This means that this version will always grow at the rate of n^2. In contrast, First-Improvement version runs in $O(n^2)$, where $O(\cdot)$ is an asymptotic upper bound, therefore First-Improvement at worst will run in n^2 but will normally run much faster. These two methods will be combined with two gain calculation methods to produce four versions of the local search method to be experimented with. In Figures 4.4 (line 11) and 4.5 (line 12), the calculation methods are placed where the `Calculate_Gain()` function is. The original gain calculation is shown in Equation 4.4 and will be referred to as the `Global Gain Calculation` (GG) as it calculates the objective value of the new solution without using any knowledge of the particular exchange being attempted.

$$Gain(f(\pi),\pi') = f(\pi') - f(\pi) \tag{4.4}$$

The Global Gain Procedure (shown in Figure 4.2) takes $\theta(mn)$ to recalculate the objective value of the new solution. This compares unfavourably with the Incremental Gain calculation whose worst-case runtime of the iterated gain version takes $O(mn)$, but as the matrix T becomes less sparse, so the runtime approaches $\Omega(m)$, where $\Omega(\cdot)$ is the asymptotic lower bound.

Figure 4.2: Pseudo-code for the Global Gain calculation

```
1  foreach actor p ∈ {1,...,m} do
2      start ← 0;
3      end ← 0;
4      for s = 1; s < n; s++ do
5          if T(p,s) then
6              start = s; break;
7          endif
8      endfor
9      for s = n; s > 0; s-- do
10         if T(p,s) then
11             end = s; break;
12         endif
13     endfor
14     for s = start + 1; s < end; s++ do
15         if T(p,s) = 0 then
16             score ← c(p) · δ(s);
17         endif
18     endfor
19 endfch
20 return score;
```

Procedure `Calculate_Gain`
input : A solution π and two pointers to shooting days i and j
output: The gain incurred from exchanging the shooting days at positions i and j.

foreach *actor* $p \in \{1,\ldots,m\}$ **do**
  $l(i) \leftarrow$ findPreviousPiecePerformed(π, i);
  $n(i) \leftarrow$ findNextPiecePerformed(π, i);
  $l(j) \leftarrow$ findPreviousPiecePerformed(π, j);
  $n(j) \leftarrow$ findNextPiecePerformed(π, j);
  if $l(i) \in [0,i)$ **then** $\hat{l}(i) =$TRUE **else** $\hat{l}(i) =$FALSE ;
  if $n(i) \in (i,j)$ **then** $\hat{n}(i) =$TRUE **else** $\hat{n}(i) =$FALSE ;
  if $l(j) \in (i,j)$ **then** $\hat{l}(j) =$TRUE **else** $\hat{l}(j) =$FALSE ;
  if $n(j) \in (j,n]$ **then** $\hat{n}(j) =$TRUE **else** $\hat{n}(j) =$FALSE ;
  $g(p) =$ Calculate gain as shown in Table 4.2;
  $TotalGain = TotalGain + g(p)$;
endfch
return $TotalGain$;

4.3 Results

To discover how efficient and useful these local search methods are they will be compared on two criteria. Firstly, the processor time taken while performing local search will be recorded and secondly, they will be compared on the quality and consistency of the best solution found. As has been described above a total of four versions will be used in these experiments.

- FI-IG: First-Improvement using the Incremental Gain calculation.
- FI-GG: First-Improvement using the Global Gain calculation.
- BI-IG: Best-Improvement using the Incremental Gain calculation.
- BI-GG: Best-Improvement using the Global Gain calculation.

These methods will be fed with solutions by a random search algorithm where the choice of which shooting day is chosen for the i-th day will be based on a uniform probability distribution. No heuristics will be used in the generation of these solutions and each iteration will be independent of its predecessors.

Figure 4.4: First-Improvement Local Search Method

input : A solution s of size n, $f(s)$ the objective value of s
input : flag_useDontLookBits : if TRUE then use Tabu bits; otherwise do not use them
output: A locally optimum solution

```
1  boolean dontLookBit[n] ← false;
2  s' ← copy(s);
3  boolean flag_improved ← true;
4  rv ← generate_random_permutation(1,n);
5  while flag_improved do
6      flag_improved ← false;
7      for l = 0;l < n;l++ do
8          loc ← rv[l];
9          if flag_useDontLookBits and dontLookBit[loc] then continue ;
10         for m = 0;m < n;m++ do
11             gain ← Calculate_Gain(loc,rv[m]);
12             if gain < 0 then
13                 loc2 ← m;
14                 break;
15             endif
16         endfor
17         if gain < 1 then
18             flag_improved ← true;
19             swap(s',loc, loc2);
20             f(s) += gain;
21             dontLookBit[loc] ← false; dontLookBit[loc2] ← false;
22         else dontLookBit[loc] ← true ;
23     endfor
24 endw
25 s ← copy(s');
26 return s;
```

Figure 4.5: Best-Improvement Local Search Method

input : A solution s of size n, $f(s)$ the objective value of s
input : flag_useDontLookBits : if TRUE then use Tabu bits; otherwise do not use them
output: A locally optimum solution

```
1  boolean dontLookBit[n] ← false;
2  s' ← copy(s);
3  boolean flag_improved ← true;
4  rv ← generate_random_permutation(1,n);
5  while flag_improved do
6      flag_improved ← false;
7      for l = 0;l < n;l++ do
8          bestloc ← 0; bestpiece ← 0; bestgain← ∞;
9          loc ← rv[l];
10         if flag_useDontLookBits and dontLookBit[loc] then continue ;
11         for m = 0;m < n;m++ do
12             gain ← Calculate_Gain(loc,rv[m]);
13             if gain < bestgain then
14                 bestloc ← m; bestgain ← gain;
15             endif
16         endfor
17         if bestgain < 1 then
18             flag_improved ← true;
19             swap(s',loc, loc2);
20             f(s) += gain;
21             dontLookBit[loc] ← false; dontLookBit[loc2] ← false;
22         else dontLookBit[loc] ← true ;
23     endfor
24 endw
25 s ← copy(s');
26 return s;
```

4.3.1 Experimental Methodology

To carry out the following set of experiments two data sets were created, one for ORSP and one for TS, using the features defined in the following table. Where the parameter required a distribution, a uniform distribution ($U(l,u)$, where l is the lowest value and u is the highest) was chosen.

Feature	Value
Actors and Shooting days ($m \times n$)	$10\times10, 15\times10, 20\times10, 30\times10,$ $10\times15, 10\times20, 10\times30$
Duration Distribution	$U(1,10)$
Cost Distribution	$U(1,10)$

Twenty problems of each combination of $m \times n$ were created, this was compounded by producing one set of ORSP instances and one set of TS instances, resulting in 280 distinct problems. The problem set was designed to be able to compare the impact of the two dimensions of the matrix T, actors m and shooting days n. Each version of the local search method was run on each problem in the data set twenty-five times, resulting in 28,000 distinct runs.

Each run consisted of 1000 iterations using a random algorithm generating the solutions that were then subjected to the particular local search method. The runs were designed to be large enough to detect the small variations involved in the time spent performing the local search method. The important items of data recorded were the following:

- total time spent over 1000 iterations in the local search method (TTLS), and
- the objective value of the best solution found during the entire 1000 iterations of the run.

4.3.2 Experiment 1: The effect of the cost distribution on performance

The rationale behind this experiment is to investigate whether the addition of the cost vector in the Talent Scheduling problem makes it harder to solve than the Orchestra Rehearsal Scheduling problem. The term *harder* is used to describe the effort the local search method has to put in to move the solution to a local optimum. If the effort is

small then the time spent performing local search will be small. On the other hand, if the local search has to do a lot of exchanges to find each improvement then the amount of time spent in the local search method will increase.

The reason why this is interesting is that, as was stated in Section 3.3.3, the ORSP has not had a formal proof confirming it to be NP-Hard and therefore it is of interest whether empirically it is easier or not. Although empirical studies cannot confirm it to be NP-Hard, it is possible to show whether or not the cost distribution makes a difference in the ability for the local search method to solve the ORSP compared to the Talent Scheduling version.

Null Hypothesis:	There is no significant difference between the means of the time taken in local search for an Orchestra Rehearsal Scheduling Problem and a Talent Scheduling Problem.
Alternative Hypothesis:	There is a difference between the means of the time taken in local search for an Orchestra Rehearsal Scheduling Problem and a Talent Scheduling Problem.

This first hypothesis tests the property that there is a performance difference when performing local search on the two data sets. The contrast would result from the variance in the cost distribution of the Talent Scheduling problems. As has already been stated a problem of the ORSP variety can be thought of as a Talent Scheduling problem with a uniform unity cost distribution. It is not expected that there will be a difference.

To test this hypothesis all the results for each problem size were grouped together into eight sets, consisting of seven distinct sets and the 10x10 set duplicated twice. This resulted in data that fitted a Normal Distribution and had similar variances, therefore the t-test was able to be used as the statistical test. Using this, seven two-tailed tests were then performed on the results produced from the two versions of local search, FI-IG and BI-IG. The results can be seen in Table 4.3.

The results show that for each set tested there is a significant difference between the means of the ORSP and TS data sets. This demonstrates that the distribution of costs does play some role in the performance of the local search method. There are two possible reasons for this:

Problem Size	BI-IG P-Values	Significant ($p \leq 0.05$)	FI-IG P-Values	Significant ($p \leq 0.05$)
10x10	< 0.0001	Yes	0.0019	Yes
15x10	< 0.0001	Yes	< 0.0001	Yes
20x10	< 0.0001	Yes	< 0.0001	Yes
30x10	< 0.0001	Yes	< 0.0001	Yes
10x10	< 0.0001	Yes	0.0019	Yes
10x15	< 0.0001	Yes	< 0.0001	Yes
10x20	< 0.0001	Yes	< 0.0001	Yes
10x30	< 0.0001	Yes	< 0.0001	Yes

Table 4.3: P-values of a t-test comparing the means of the time spent in local search of the ORSP data set against that of the TS data set. (Bonferroni corrected α is $\frac{0.05}{8} = 0.00625$)

- the method performs a lower number of exchanges improving the solution for ORSP than TS, or
- the method performs a lower number of exchanges prior to an improvement for ORSP than TS.

Whatever the reason, the difference between the two sets is small as can be seen in Figure 4.6. This set of four figures shows that, where there is a difference, the Talent Scheduling results take longer and this difference increases as either m or n increases. Therefore it is most appropriate to perform one-tailed tests on any future experiments using the hypothesis that the ORSP data set will be significantly less than the TS data set.

To test which of these two factors, or a combination of them, could explain this result, the mean number of exchanges resulting in an improvement per run of one thousand local searches was recorded for each data set, as well as the average number of exchanges tested before an improvement was found. For this experiment the same data sets and seeds were used. A summary of the results can be seen in Tables 4.4 and 4.5.

A possible alternative would have been to add a random cost vector to the existing ORSP problems. The disadvantage of this would be that there may have been a bias towards the creation of the ORSP problems favouring unit vectors. Therefore this was not done, instead a large sample of varying problems were generated for each set (in

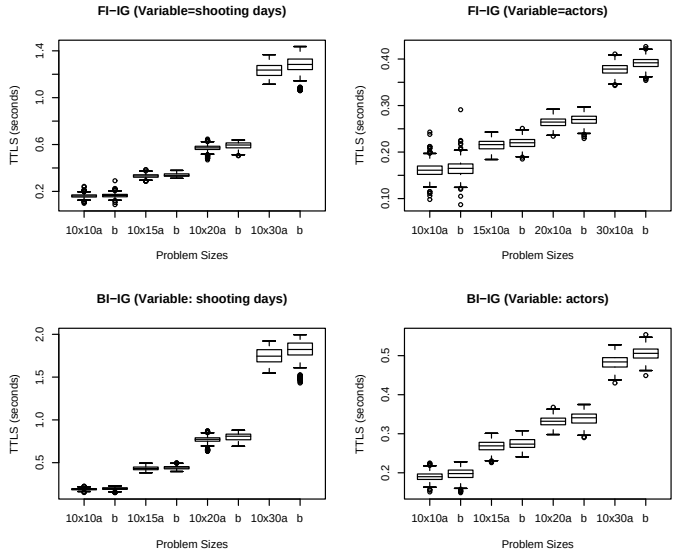


Figure 4.6: Boxplots depicting the distribution of times spent in local search over all problems of each problem size for each data set. (Notation: actors x shooting days, a=ORSP data set, b=TS data set)

this case 20 problems of each size). By doing this the probability of the bias of any one particular problem being able to skew the results was minimised.

Table 4.4 shows the number of exchanges looked at before an improvement is accepted. The first observation is that the differences are not very large as the numbers are very close to unity. Using a t-test, a comparison was made for each size of problem between the two data sets. All of the tests rejected the Null Hypothesis, therefore the mean of the ORSP data set was significantly less than the mean of the TS data set.

In Table 4.5 the number of exchanges that resulted in improvements to the solution is given. This reveals once more the results are close to unity and that the number of improvements in the ORSP data set tends to be less than that of the TS data set. Again, using the t-test, it was found that the Null Hypothesis was rejected, therefore the conclusion was that the mean of the ORSP data set was significantly less than the mean of the TS data set.

Both of these sets of results point to a combination of factors resulting in the local search performing faster on those problems with unit cost vectors. The reason why the local search methods works faster on the ORSP set than the TS data set is a combination of less exchanges being required to be tested and less improvements being made per local search call.

FI-IG							
m x *n*	Min.	1st Q.	Median	Mean	St. Dev.	3rd Q.	Max.
10x10	0.92	0.95	0.96	0.97	0.03	0.98	1.02
10x15	0.94	0.97	0.98	0.98	0.03	1.01	1.05
10x20	0.87	0.96	0.98	0.97	0.03	0.99	1.01
10x30	0.92	0.97	0.98	0.98	0.03	1.00	1.04
10x10	0.92	0.95	0.96	0.97	0.03	0.98	1.02
15x10	0.91	0.95	0.98	0.97	0.03	1.00	1.02
20x10	0.94	0.98	0.99	0.99	0.02	0.99	1.03
30x10	0.93	0.97	0.97	0.98	0.02	1.00	1.01
BI-IG							
10x10	0.91	0.93	0.96	0.96	0.04	0.99	1.02
10x15	0.9	0.95	0.99	0.99	0.05	1.02	1.06
10x20	0.91	0.95	0.96	0.97	0.03	0.99	1.05
10x30	0.92	0.94	0.97	0.98	0.06	1.00	1.16
10x10	0.91	0.93	0.96	0.96	0.04	0.99	1.02
15x10	0.87	0.93	0.97	0.97	0.05	1.00	1.06
20x10	0.92	0.96	0.99	0.98	0.04	1.01	1.04
30x10	0.9	0.95	0.98	0.97	0.04	1.00	1.04

Table 4.4: Summary table of results showing the ratio of number of exchanges tested per improvement for the ORSP data set against that of the TS data set. (Q=Quartile, $\frac{\text{ORSP data set}}{\text{TS data set}}$)

FI-IG							
m x *n*	Min.	1st Q.	Median	Mean	St. Dev.	3rd Q.	Max.
10x10	0.83	0.88	0.94	0.93	0.06	0.98	1.02
10x15	0.84	0.94	0.97	0.97	0.06	1.01	1.05
10x20	0.86	0.9	0.93	0.93	0.05	0.95	1.07
10x30	0.87	0.91	0.92	0.94	0.06	0.95	1.14
10x10	0.83	0.88	0.94	0.93	0.06	0.98	1.02
15x10	0.84	0.9	0.95	0.95	0.07	0.98	1.07
20x10	0.9	0.94	0.99	0.98	0.05	1.01	1.06
30x10	0.86	0.91	0.97	0.96	0.05	0.99	1.03
BI-IG							
10x10	0.84	0.89	0.94	0.94	0.06	0.98	1.04
10x15	0.89	0.96	0.99	0.98	0.04	1.02	1.04
10x20	0.92	0.94	0.96	0.96	0.02	0.97	1.02
10x30	0.92	0.95	0.95	0.96	0.03	0.97	1.06
10x10	0.84	0.89	0.94	0.94	0.06	0.98	1.04
15x10	0.84	0.91	0.96	0.96	0.08	1.01	1.11
20x10	0.88	0.95	0.99	0.98	0.05	1.02	1.06
30x10	0.84	0.94	0.98	0.96	0.05	1	1.04

Table 4.5: Summary table of results showing the ratio of the number of improvements made per local search. (Q=Quartile, $\frac{\text{ORSP data set}}{\text{TS data set}}$)

4.3.3 Experiment 2: To investigate the difference between the two gain calculation methods

This second experiment provides evidence for the improvement in performance that using the Incremental Gain method should make over the use of the Global Gain calculation. This difference will emerge from comparing the total time spent performing local search for the two data sets.

It is expected that the Global Gain calculation should take a total processing time of the order $\theta(nm)$. The Incremental Gain version on the other hand only looks at those actors affected by the two shooting days being exchanged and therefore should of the order $O(nm)$. The expected improvement in performance from the Global to Incremental Gain calculation should be a factor of between 1 and n for both First- and Best-Improvement methods.

The hypothesis being tested in this subsection is the following:

Null Hypothesis	There will be no significant difference in the means of the distributions of TTLS between FI-IG and FI-GG when run on either data set.
Alternative Hypothesis	FI-IG TTLS will be significantly lower than those of FI-GG.

An analogous hypothesis test for the Best-Improvement version is also tested.

Tables 4.6 and 4.7 gives a summary of the results. The first test that was performed was to discover whether the ratio of First-Improvement methods was the same as that of the Best-Improvement methods. To perform this test the means of the ratios were compared using a t-test. As expected there was no significant difference between the ratios of the Global to Incremental gain times for Best-Improvement and First-Improvement methods for either ORSP or TS.

There are two interesting observations that can be made about these results. The first is that there is a decrease in the ratio of those sets where m is increasing. This indicates that as the number of actors is increased the Global Gain and Incremental Gain both are converging towards 1. The reason for this is that the Incremental Gain is increasing faster than Global Gain.

A similar observation can be made about the increase in the ratio of those sets where n is increasing. In this case the Global Gain is increasing faster than the Incremental Gain. Both these observation demonstrate the dominance of each factor in the runtime. If one has a large m and small n then the Global Calculation may result in very little difference in performance, but if the reverse is true then it is worth using the Incremental Gain for the increase in performance.

The second observation to make about these two tables is that the standard deviation in the ratio of the means is very low around 0.1. This indicates that the trends described in the previous paragraph are reliable and for a random problem one would not expect large deviations from these performance ratios.

To test the hypothesis at the start of this experiment, the mean of the time spent in local search for each of the twenty problems was calculated. A t-test was then performed on each of the 7 data sets for each problem. All the results showed that the Null Hypothesis should be rejected. Therefore, it was shown that the Incremental Gain calculation took less time than the Global Gain calculation as predicted.

4.3.4 Experiment 3: The effect of the number of actors and shootings days on the local search

The primary two dimensions of these type of problems is the number of actors and the number of shooting days. Therefore, an interesting question is how the local search scales with respect to an increase in each of these dimensions. To investigate this the problem sets were designed to increase each parameter equally from a starting size of 10 by 10. By plotting these results on appropriate axis it would then be possible to answer this question.

Figure 4.7 shows two graphs. The left graph shows how the time spent in local search grows with the increase in the number of actors. The mean of each problem size has been plotted and a regression line has been plotted. The graph demonstrates how for both local search methods employing the Incremental Gain calculation and on both data sets, the growth is approximately the same. The gradient of the regression line is approximately 0.01, which means that for every actor a 0.01 increase in the total time spent in local search would be anticipated.

The graph on the right of the figure plots the same information but for the number

$\frac{FI-GG}{FI-IG}$ **Summary Table**							
m x *n*	Min.	1st Q.	Median	Mean	St. Dev.	3rd Q.	Max.
10x10	3.50	3.61	3.64	3.64	0.08	3.67	3.85
15x10	2.95	3.04	3.09	3.10	0.09	3.15	3.28
20x10	2.72	2.73	2.77	2.78	0.05	2.82	2.89
30x10	2.31	2.33	2.34	2.35	0.03	2.37	2.39
10x10	3.49	3.59	3.64	3.64	0.08	3.67	3.87
10x15	4.40	4.54	4.59	4.60	0.12	4.65	4.84
10x20	4.77	5.00	5.08	5.05	0.11	5.12	5.24
10x30	5.74	5.86	5.93	5.93	0.12	6.02	6.12

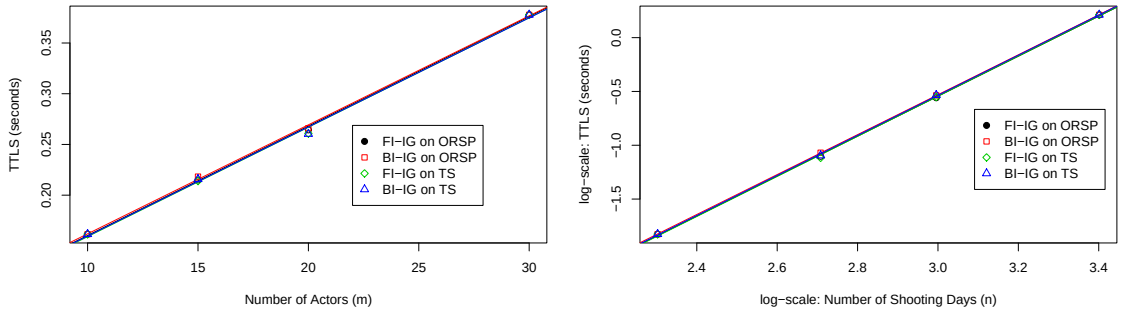
$\frac{BI-GG}{BI-IG}$ **Summary Table**							
m x *n*	Min.	1st Q.	Median	Mean	St. Dev.	3rd Q.	Max.
10x10	3.86	3.97	4.00	4.03	0.09	4.10	4.19
15x10	3.14	3.24	3.26	3.27	0.09	3.31	3.50
20x10	2.81	2.85	2.89	2.89	0.06	2.93	3.00
30x10	2.36	2.40	2.41	2.42	0.04	2.44	2.50
10x10	3.86	3.98	4.01	4.04	0.10	4.11	4.22
10x15	4.57	4.74	4.78	4.79	0.12	4.86	5.00
10x20	4.93	5.18	5.24	5.23	0.12	5.30	5.46
10x30	5.86	5.98	6.03	6.04	0.11	6.12	6.22

Table 4.6: Summary for the ORSP data set comparing the Incremental Gain (IG) calculation to the Global Gain (GG) calculation. (Q=Quartile, St.Dev.=Standard Deviation)

$\frac{FI-GG}{FI-IG}$ **Summary Table**							
m x *n*	Min	1st Q.	Median	Mean	St. Dev.	3rd Q.	Max
10x10	3.61	3.67	3.74	3.78	0.12	3.83	4.06
15x10	3.11	3.18	3.20	3.22	0.06	3.26	3.35
20x10	2.77	2.86	2.89	2.89	0.07	2.92	3.00
30x10	2.33	2.41	2.43	2.44	0.05	2.46	2.54
10x10	3.61	3.67	3.74	3.78	0.12	3.83	4.06
10x15	4.52	4.69	4.74	4.75	0.11	4.82	4.97
10x20	5.09	5.22	5.29	5.28	0.11	5.33	5.56
10x30	6.08	6.19	6.29	6.28	0.14	6.37	6.63

$\frac{BI-GG}{BI-IG}$ **Summary Table**							
m x *n*	Min	1st Q.	Median	Mean	St. Dev.	3rd Q.	Max
10x10	3.90	4.03	4.15	4.15	0.15	4.24	4.58
15x10	3.25	3.33	3.41	3.39	0.08	3.45	3.52
20x10	2.88	2.99	3.02	3.03	0.08	3.09	3.16
30x10	2.41	2.47	2.50	2.51	0.06	2.54	2.62
10x10	3.90	4.03	4.15	4.15	0.15	4.24	4.58
10x15	4.79	4.92	5.02	4.99	0.11	5.06	5.22
10x20	5.27	5.38	5.47	5.47	0.11	5.53	5.68
10x30	6.19	6.29	6.37	6.39	0.14	6.48	6.75

Table 4.7: Summary Table for TS data set comparing the Incremental Gain (IG) calculation to the Global Gain (GG) calculation. (Q=Quartile, St.Dev.=Standard Deviation)



(a) Increasing m (b) Increasing n

Figure 4.7: Graphs illustrating the growth of the time spent in the local search (TTLS) for FI-IG and BI-IG on the two data sets with the increase in (a) the number of actors and (b) the number of shooting days.

of shooting days. This increase is clearly an exponential increase, as it is plotted on log-log axes. Again the regression line has been plotted and once more the difference between the various versions is small. The gradients of the regression lines are in the range $[1.86, 2.03]$, where the higher gradients are for the Best-Improvement versions.

These results are not unexpected and is confirmation of the analysis given earlier in the chapter that indicated the dominant factor would be the number of shooting days.

4.3.5 Experiment 4: Comparison of the TTLS between First- and Best-Improvement methods

This experiment compares the difference in the total amount of time spent in local search between the First-Improvement method and the Best-Improvement method for both data sets. The expected outcome is that the First-Improvement method will spend less time performing the local search due to the reduced number of exchanges tested.

Null Hypothesis:	There is no difference between the mean of the TTLS of the First-Improvement method and the mean of the TTLS of the Best-Improvement method.
Alternative Hypothesis:	The mean of the TTLS of the First-Improvement method is less than the mean of the Best-Improvement method.

To test this hypothesis a t-test was performed on each of the eight problem sizes for the two data sets. All the significance tests showed that the Null Hypothesis should be rejected and the Alternative Hypothesis should be accepted. This is not very surprising given what has already been illustrated in earlier experiments.

In Table 4.8 the two methods are compared as a ratio. This is important to determine the performance increase that can be obtained by using FI-IG. The table shows that by using FI-IG one can receive an approximate 25% increase in performance when increasing the number of actors. When increasing the number of shooting days this performance increase is more valuable becoming 40% or more. The table also shows that these increases are constant, with a standard deviation of approximately 0.02.

To test these regressions, twenty large problem were created of size 10x100 and 100x10. These were then approximated using the two methods. In Table 4.9 the observed values are compared to the expected values from the regression lines. The percentage error is calculated as follows: $\frac{O-E}{E} * 100$, where O is the observed mean and E is the expected mean.

The expected values for the First-Improvement method are not too inaccurate, although the Best-Improvement values are quite a long way out. This is due to the number of sets used, to get a more accurate estimate more problem sizes should be used. However, these values do demonstrate the performance level that can be gained if the First-Improvement method is used, which in problems of this size the improvement is approximately 30% for both data sets.

Although the performance of the First-Improvement local search is better in terms of time spent in local search, the next question is how does this affect the quality of the solutions being generated.

$\frac{BI-IG}{FI-IG}$ **Summary Table for ORSP**							
m x *n*	Min	1st Q.	Median	Mean	St. Dev.	3rd Q.	Max
10x10	1.13	1.16	1.17	1.18	0.03	1.20	1.22
15x10	1.20	1.23	1.24	1.24	0.02	1.26	1.27
20x10	1.24	1.24	1.25	1.26	0.01	1.26	1.29
30x10	1.26	1.27	1.28	1.28	0.02	1.29	1.31
10x10	1.11	1.16	1.18	1.18	0.03	1.20	1.23
10x15	1.28	1.30	1.31	1.31	0.02	1.32	1.33
10x20	1.31	1.34	1.35	1.35	0.02	1.35	1.37
10x30	1.39	1.40	1.41	1.41	0.02	1.42	1.44
$\frac{BI-IG}{FI-IG}$ **Summary Table for TS**							
m x *n*	Min	1st Q.	Median	Mean	St. Dev.	3rd Q.	Max
10x10	1.13	1.18	1.20	1.19	0.03	1.21	1.23
15x10	1.22	1.23	1.25	1.25	0.02	1.26	1.28
20x10	1.23	1.25	1.26	1.26	0.01	1.27	1.28
30x10	1.27	1.28	1.30	1.29	0.02	1.30	1.33
10x10	1.13	1.18	1.20	1.19	0.03	1.21	1.23
10x15	1.28	1.30	1.30	1.30	0.01	1.31	1.33
10x20	1.33	1.34	1.36	1.36	0.01	1.36	1.38
10x30	1.38	1.42	1.42	1.42	0.01	1.43	1.44

Table 4.8: Summary table of results showing the ratio of BI-IG to FI-IG for the total time spent in local search. (Q=Quartile, St.Dev.=Standard Deviation)

FI-IG						
	ORSP Data Set			TS Data Set		
m x *n*	Expected Mean	Observed Mean	% Error	Expected Mean	Observed Mean	% Error
100x10	1.1	1.1	4.8	1.1	1.1	6.3
10x100	11.5	12.4	7.5	11.8	12.3	3.9
BI-IG						
100x10	1.1	1.5	28.6	2.0	1.5	-26.5
10x100	19.9	17.6	-13.2	21.1	18.0	-17.4

Table 4.9: Table showing the expected and observed values for large ORSP and TS problems.

4.3.6 Experiment 5: Comparison of the quality of solutions produced by First- and Best-Improvement methods

In Experiment 4 it was shown that there are reliable large increases in performance by using FI-IG over BI-IG, especially as n increases. In this experiment the question of solution quality is studied to see if this increased performance comes at a price.

Null Hypothesis: There is no difference between the mean of the solution quality of First-Improvement and the mean of the solution quality of the Best-Improvement.

Alternative Hypothesis: The mean of the solution quality of the First-Improvement method is greater than the mean of the Best-Improvement.

To test this hypothesis the quality of the solutions was compared between the FI-IG and BI-IG methods on the two data sets. For these experiments the t-test was used to check for significance for each of the problem sets. For each of the twenty problems per size, the mean of the 25 samples was calculated and a paired t-test was then performed for each problem size. These tests showed that there was no difference between the quality of the two methods.

From the original results, the First-Improvement method was only significantly worse when the number of shooting days was equal to 30 and 100. Therefore from this result and the previous results for problems of this size the First-Improvement method should be used to harness the gain in performance.

The results in Table 4.10 summarise the differences in the quality of the best solutions produced by the First-Improvement and Best-Improvement methods. What is immediately clear is that there is nothing to tell them apart in small problems where the variation is only in the number of actors. When the number of pieces is increased the First-Improvement can still produce solutions of the same quality as the Best-Improvement method but less consistently.

Where BI-IG does get better solutions the average percentage difference is only 0.6% for the ORSP data set and 0.94% for the TS data set. As with the previous experiment the two larger sets (100x10 and 10x100) were compared for quality. In the 100x10 set

ORSP Data Set				**TS Data Set**		
Problem Size	Min.	Mean	Max.	Min.	Mean	Max.
10x10	100	100	100	100	100	100
15x10	100	100	100	100	100	100
20x10	100	100	100	100	100	100
30x10	100	100	100	100	100	100
10x10	100	100	100	100	100	100
10x15	100	100	100	100	100	100
10x20	100	45	65	100	20	30
		(0/55)	(0/35)		(0/80)	(0/70)
10x30	90	0	0	65	0	0
	(0/10)	(0/100)	(0/100)	(0/35)	(0/100)	(0/100)

Table 4.10: Summary of the difference in quality of solutions produced by the First-Improvement (FI) and Best-Improvement (BI) methods. (Percentage notation: FI=BI (FI<BI / FI>BI). The equality is the main number, signifying the number of times the two improvement methods gave the same quality of solution. The other two identify the number of times one creates better solutions (has a smaller objective value) than the other.)

FI-IG and BI-IG on both ORSP and TS data sets managed to achieve the same quality of solution, but on the problems of size 10x100 the differences were more noticeable. For both data sets the FI-IG method failed to find a solution of the same quality as the BI-IG and had an average percentage difference of 4.15% for the ORSP data set and 7.12% for the TS data set. Although these differences are larger, with a good meta-heuristic this difference should not have a significant impact.

4.3.7 Experiment 6: Performance of FI-IG and BI-IG on the existing problems

This final experiment is to report on the performance of the local search method on the three problems that have been seen before in [Smith, 2003; Gregory et al., 2004]. The interesting question is whether the local search will be able to solve all three, or whether like the TSP and QAP, there is one that requires a more sophisticated approach.

To perform the experiment the local search method was run 25 times on each problem for both methods. Each run was given a maximum of 5000 iterations, which meant that the local search acted on 5000 separate random solutions, the best of which was then taken as final solution. The number of iterations was increased from 1000 to give the local search the best chance of finding the known optimal solution. The results produced are summarised in Table 4.11.

From the results one can see that the local search performed well for all three problems. The reason for FI-IG taking slightly longer than BI-IG is due to the greater number of iterations required to find an optimal solution.

4.4 Conclusions

In this chapter six experiments were done to investigate how four local search methods would compare against each other in terms of time spent performing the local search, and in the quality of the solutions produced. The versions tested were:

- FI-IG: First-Improvement using Incremental Gain calculation,
- BI-IG: Best-Improvement using Incremental Gain calculation,
- FI-GG: First-Improvement using Global Gain calculation,

FI-IG Local Search							
Instance ($m \times n$)	**Min.**	**1st Q.**	**Median**	**Mean**	**St. Dev.**	**3rd Q.**	**Max.**
ORSP (5x9)							
% above opt	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Time	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Iterations	1.0	3.0	5.0	5.2	2.4	7.0	9.0
TS 1 (10x13)							
% above opt	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Time	0.00	0.01	0.02	0.02	0.02	0.03	0.08
Iterations	2.0	11.0	33.0	42.0	38.6	57.0	160.0
TS 2 (8x20)							
% above opt	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Time	0.05	0.11	0.26	0.37	0.39	0.50	1.85
Iterations	46.0	114.0	265.0	384.0	403.4	513.0	1893.0
BI-IG Local Search							
ORSP (5x9)							
% above opt	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Time	0	0	0	0	0	0	0
Iterations	1.0	2.0	3.0	4.2	3.3	6.0	12.0
TS 1 (10x13)							
% above opt	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Time	0.00	0.01	0.01	0.02	0.01	0.02	0.06
Iterations	1.0	8.0	19.0	26.8	24.8	38.0	98.0
TS 2 (8x20)							
% above opt	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Time	0.02	0.16	0.23	0.36	0.30	0.44	1.05
Iterations	14.0	130.0	194.0	311.1	258.0	379.0	903.0

Table 4.11: Results for solving existing problems using FI-IG and BI-IG. (Q=Quartile. The percentage above optimum is given to one decimal place, and the time taken to find best solution in brackets is given to two decimal places.)

- BI-GG: Best-Improvement using Global Gain calculation.

From these six experiments the following conclusions were drawn.

1. The distribution of the cost vector values does significantly affect the performance of the local search method.
2. The Increment Gain calculation is significantly faster than the Global Gain method.
3. As the number of shooting days is increased the performance gain from using Incremental Gain increases.
4. Increasing the number of actors, leads to linear growth in the performance of the Incremental Gain.
5. Increasing the number of shooting days, leads to exponential growth in the performance of Incremental Gain.
6. Using the First-Improvement method produces an approximate 30% increase in performance over the Best-Improvement.
7. The growth of First- and Best-Improvement methods is approximately the same for increasing the number of actors and increasing the number of shooting days.
8. The increase in performance comes at a price of losing approximately between 4% and 7% on the quality of the solution for large problems (the number of shooting days $= 100$); for the number of shooting days less than 30 there is no noticeable difference and solutions of the same quality can still be obtained using both methods.

There are a number of experiments that would be of benefit in the understanding of how this problem scales. These include varying the distribution of values in the duration vector and doing a more controlled study of how the cost vector affects the scalability and performance of the local search. For these experiments these distributions were fixed as a uniform distribution in the range $[1, 10]$ but as the scale grows and the distribution increased perhaps the difficulty could be kept at a manageable level. As a follow-up study it would also be of interest to see how these two vectors interact in restricting possible choices of schedules.

As an extension to Experiment 2, it should also be possible to correlate the distribution of ones in the matrix T to the performance of each local search. This would confirm which local search method was better for varying levels of sparseness in the matrix.

A Branch-and-Bound algorithm, based on the study by [Cheng et al., 1993], was also implemented to find optima for the problems created for this study. Unfortunately the algorithm was too slow and could not handle problems larger than ten shooting days. The problem was that the bound used in this earlier study could not distinguish between early branches, therefore the search space was not able to be pruned sufficiently to enable a complete search to take place. This is another area where more work could be done. This would provide an optimum to compare the quality of the solutions produced by the local search, which would be helpful in assessing the quality of the method.

4.5 Summary

In this chapter a local search method for use with the ORSP and Talent Scheduling problems has been developed. A number of versions were implemented and it was found that the incremental version performed as expected, being much faster than using the full objective function. Also First-Improvement and Best-Improvement strategies were implemented and it was shown that there was little difference in the ability of both to find solutions of similar quality and that the First-Improvement performed faster than Best-Improvement by up to 40%. For the purposes of the experiments later on in this thesis, for the reasons illustrated in this chapter, the local search that will be used will be the First-Improvement Incremental Gain Local Search.

Part II

Contributions

Part I of the thesis has described all the necessary background information required for an understanding of the area on which the main objectives of this thesis are focused. Chapter 1.2 explained the basic parts of Graph Theory required to understand how the algorithms solve various combinatorial optimisation problems, while Chapter 2 reviewed the many types of Ant algorithms and some of their competitors. At the end of that chapter a number of criticisms were raised and it will be the role of Part II to expand upon and hopefully answer a number of these.

In Chapter 3 the application and problems were introduced alongside some basic methodology that will make the conclusions more rigorous. Afterward in Chapter 4 a local search was described for the Talent Scheduling problem which will be used in Chapters 7 and 8.

The objectives that will be achieved in the next four chapters are the following:

- *Chapter 5 - An Empirical Study of the Graph-Based Ant System Model*: The first objective of this thesis is to present an implementation of the Graph-based Ant System (GBAS) that was introduced in Subsection 2.2.10. The reason for this is that it is the only theoretical version that exists of an Ant Algorithm for a graph representation; with the advantage that it does not have problem specific modifications. There is no current implementation of this algorithm and therefore this will be an original implementation. Ant System and Max-Min Ant System are used as two examples of current implementations to give an idea of the quality of GBAS as a model. The main objectives will be to investigate how the parameters affect GBAS and how these correlate to the other implementations.

  The experiments for this chapter are:

  1. Variation in ρ (Section 5.4, page 173) explores how varying the decay factor affects the three algorithms. The primary conclusion is that setting $\rho > 0.9$ reduces the ability of the GBAS model to predict the other algorithms.

  2. Variation in α, β (Section 5.5, page 181) explores how varying the influence of both the pheromone matrix and the heuristic affects the three algorithms. The conclusion was that high α makes the model less applicable, while high β makes the model more so.

  3. Variation in m (Section 5.6, page 193) explores how varying the number of ants used per iteration affects the three algorithms. This experiment concluded that,

for all three algorithms, no more than 10 ants were required for good performance, although as $\rho \rightarrow 0$, m should be increased.

Therefore the final conclusion of this chapter is that, in general, for parameters settings which would normally be chosen by a researcher, the GBAS model is consistent with the behaviour of other Ant algorithms, such as AS and MMAS.

- *Chapter 6 - Heuristics as a Source of Knowledge in Ant Algorithms*: The second objective is to demonstrate how knowledge that is introduced via a heuristic is used and how the quality of this knowledge affects the performance of the algorithm. Heuristics are very rarely justified in terms of quality and their effect on Ant algorithms has not been widely studied. In this chapter five heuristics are used to show how varying degrees of misinformation affect the algorithms.

  The experiments for this chapter are:

  1. Sub-section 6.3.1 (page 217) describes the properties of each individual heuristic. It shows how the three algorithms react to the heuristics in terms of the quality of solutions and their convergence as α and β are varied.

  2. Sub-section 6.3.2 (page 241) explores how each algorithm differs from the performance of the Nearest Neighbour heuristic. This analysis enables the identification of traits in the misleading heuristics that may perform better than in their more accurate counterpart.

  3. Sub-section 6.3.3 (page 264) checks explicitly the claim that dynamic heuristics produce better quality solutions, and later convergence, than static heuristics. The conclusion is that given the heuristic is not simply an increment to the current cost distribution, the dynamic heuristic is shown to have these properties.

  Therefore the single major conclusion of this chapter is that it shows misleading heuristics greatly affect the performance of the algorithm. Therefore, this explains why in many applications no heuristic is better than a misleading heuristic. Thus, heuristics should be used with care, and more intelligently, with the Ant algorithms.

- *Chapter 7 - An Empirical Investigation of Ant Algorithms with Local Search*: The third objective is to combine local search methods with the three algorithms from Chapter 5, which were GBAS, Ant System and Max-Min Ant System, to see how it affects their performance. The chapter concentrates on how the local search

method impacts on the choice of parameters and on the behaviour of the three algorithms.

The experiments for this chapter are:

1. Variation in ρ (Section 7.3, page 277) explores how varying the decay factor affects the three algorithms. The primary conclusion is that many respects the behaviour is similar to that in shown without the use of local search. However, local search methods do make the behaviour of the algorithms more dependent upon the specific attributes of the problem, and the strength of GBAS to correlat with the other two algorithms is diminished.

2. Variation in α, β (Section 7.4, page 285) explores how varying the influence of both the pheromone matrix and the heuristic affects the three algorithms. There are two main conclusions from this experiment. Firstly, increases in both α and β make the algorithms more independent. Secondly, β becomes less influential on the final solution quality, as the strength of the local search method increases.

3. Variation in m (Section 7.5, page 302) explores how varying the number of ants used per iteration affects the three algorithms. This experiment concluded that, for all three algorithms, no more than a single ants were required for good performance, although as $\rho \rightarrow 0$, m should be increased.

Therefore, this chapter showed that not only the inclusion, but the strength, of a local search method has a direct effect on the behaviour of the three algorithms. It showed that local search makes the algorithms more independent, and their behaviour more dependent on the problem instance.

- *Chapter 8 - Understanding the role of Ant Algorithms when combined with Local Search*: The fourth and final chapter of Part II takes a step back and investigates the reasons for combining an Ant algorithm with a local search method. It looks at how simple fixed and variable neighbourhood searches can act as effective drivers for local search and tries to identify what role Ant algorithms are playing when combined with a local search method.

  This chapter showed that the reason why the Max-Min Ant System is more effective than the other two Ant algorithms is because it offers the greatest variation of solutions to seed the local search method. It concludes that hybrid algorithm could be used more effectively if greater variation was achieved in the solutions seeding

the local search.

Chapter 9 then brings together Part I and Part II, detailing a synopsis of the main conclusions of the thesis. The chapter also discusses how the work affects current research and how this work could be taken further in the future.

Chapter 5

An Empirical Study of the Graph-Based Ant System Model

This chapter will answer the question "Does the theoretical Graph-based Ant System (GBAS) model empirically represent the characteristics of implemented Ant algorithms?" To answer this question this chapter describes the Graph-based Ant System algorithm introduced in [Gutjahr, 2000, 2003b], and investigates how well it models other influential algorithms such as Ant System and the Max-Min Ant System. To this end five experiments are described that test how all three algorithms behave when their common parameters are varied.

In Section 5.1 the motivation for implementing this model will be explained, followed by the details of the algorithm in Section 5.2. Section 5.3 will describe the common attributes of the experiments which follow in Sections 5.4 to 5.7. The chapter concludes with a discussion of the main findings in Section 5.8.

5.1 Introduction

In the review of Ant algorithms in Chapter 2 the only algorithm that had a rigorous theoretical background is the Graph-Based Ant System (GBAS), set out for the first time in [Gutjahr, 2000] and generalised in [Gutjahr, 2003b]. Unfortunately this algorithm has not been implemented, therefore the application of the theoretical model has not been tested. A worst case scenario is assumed by the model to achieve its proofs, and there remains the question of whether this is a good model for discussing current

algorithms that have already been implemented. Therefore, the following assumptions will be used which will justify the comparison, and make explicit the assumption that GBAS is the benchmark algorithm for any expected results:

Assumption: GBAS is an optimising algorithm. Assumption: The behaviour of GBAS is representative of other Ant algorithms in their most basic form.

The first assumption is clearly required, however the second is more controversial. It claims that how the parameters modify the behaviour of the algorithm in the GBAS model is how AS and MMAS intended their parameters to behave. Therefore, it can be expected that AS and MMAS should behave like GBAS.

Ant System was chosen as it was the first major implementation of an Ant algorithm and many of the subsequent algorithms are descendents of this algorithm. Therefore, by using this algorithm it is likely that the conclusions reached will be applicable to these later algorithms. Max-Min Ant System was chosen because it was claimed in [Stützle and Dorigo, 2002] that the convergence proof for GBAS would not work for MMAS as they were too different. By including this algorithm it will be shown whether their behaviour is empirically different.

This chapter focuses on how well the model correlates to Ant System (AS) and Max-Min Ant System (MMAS) by varying their common parameters in suitable ranges. These two algorithms were chosen as both form the basis for most of the implementations in this research area, thus maximising the application of this work. To achieve this, AS and MMAS have been stripped down to their essential components. Therefore in these experiments there is:

- no nearest neighbour parameter (*nn*) to reduce the search space,
- no parameters, such as q_0, to control exploration versus exploitation,
- no local search method,
- and no random restarts.

The parameters *nn* and q_0 were added to the original algorithms to achieve greater success in the early applications. Both of these parameters were removed as each distorts the search space in favour of the algorithm with unpredictable effects.

nn reduces the number of possible choices at each node in the construction graph to achieve two goals. The first goal is to reduce the search space so that the algorithm

finds better solutions faster. The second was to speed up the construction process for large problems, which grows significantly as $n \to \infty$.

q_0 enables the algorithm to choose the next node in the construction graph either with a random distribution, or for the choice to rely on the heuristic and pheromone information. Therefore q_0 introduces an extra element of randomness which enables convergence to be delayed and allows greater diversity of the search. The absence of this parameter will be compensated for by setting the other parameters appropriately.

Local search methods and random restarts have been removed to make the algorithms as simple as possible so that the effects of the parameter variations can be clearly detected. These two techniques are generic and can be applied to any metaheuristic and are therefore removed to see how effective Ant algorithms are acting alone.

The algorithms will have the following common parameters:

- α, to weight the influence of the pheromone matrix,
- β, to weight the influence of any heuristics used,
- ρ, to regulate the pheromone decay,
- m, the number of ants per iteration,
- max_{it}, the maximum number of iterations the algorithm will run for,
- max_t, the maximum amount of time to run each algorithm for,
- $\Delta\tau_{ij} = \frac{Q}{1+f(s)}$, where $Q = 1$, as the amount of pheromone to add in accordance with the update rule,
- only the global best solution(s) will be used to update the pheromone matrix in accordance with the particular algorithm.
- $\tau_0 = 5$ for AS, $\tau_0 = \frac{1}{n^2}$ for GBAS, $\tau_0 = 3$ for MMAS.
- MMAS specific parameters: $p_{best} = 0.05$,$\tau_{max} = 3$ or if the optimum is known τ_{max} is set to this value instead. The pheromone limits, τ_{min} and τ_{max}, will be adjusted after $t = 0$ so their initial values are not significant as long as they are greater than zero.

These parameters have been chosen to make the algorithms as similar as possible, exposing the primary differences which lie in the pheromone update rule and the matrix representation. From the exploratory experiments the parameters max_{it} and max_t were

given appropriate values. The parameter max_{it} was set to 500 for the majority of experiments and $\frac{1000}{m}$ for experiments in Section 5.7, where m is varied to control the total number of samples. The value of 500 was used as both GBAS and AS converged before this value on the chosen problems. MMAS does take longer to converge occasionally, but as the focus is on comparing it to GBAS this value was chosen as a compromise between experimental length and quality of the solutions achieved per run. The same reasoning goes for setting the maximum amount of time to five minutes, which was found to be ample time for all the algorithms to complete 500 iterations.

$\Delta\tau_{ij}$ is normally set in AS and MMAS so that Q is equal to 100, but to keep conditions between the algorithms similar the value was changed to 1. This means that the increments are *smaller* than in other applications. However, this is not a significant as the modification only delays convergence and makes the algorithms less greedy. Furthermore, when updating the pheromone matrix only the global best solution is used, again to keep as the conditions as similar as possible. Various researchers have tried alternating between using the iteration best and the global best to update the pheromone matrix in an effort to increase the diversity and delay convergence with mixed success. In the convergence proof for GBAS it does not require any such modification, therefore it was deemed an unnecessary complexity and was not included as one the test parameters.

τ_0 is the initial amount of pheromone intensity for each arc in the construction graph. In all three it has been set to values quoted in previous AS, MMAS and GBAS papers. In addition, p_{best} and τ_{max} have to be specified for MMAS and again these were set to values that have been applied to the domains of TSP and QAP respectively [Stützle and Hoos, 2000]. The value for τ_{max} at the start of a run is irrelevant as it is altered immediately after the first set of solutions has been constructed. These two supplementary parameters have not been varied as they are not common between all the algorithms, but all are good first effort values that a researcher might try before optimising the algorithm for their particular domain.

Assumption: The parameter τ_0 is not fundamental to the behaviour of AS or MMAS algorithm if $\tau_{ij} = \tau_0 \; \forall i, j$ and $\tau_0 > 0$.
Assumption: The parameters p_{best} and τ_{max} that are specific to MMAS do not alter the general performance of the algorithm significantly.

The rest of the chapter consists of the theory behind the Graph-Based Ant System, followed by an overview of the experiments carried out. Then the method and results for each experiment will be discussed and finally the conclusions will be drawn at the end of the chapter.

5.2 Description of the Graph-Based Ant System

This section describes the Graph-Based Ant System algorithm (GBAS). The description has been paraphrased from the original publications [Gutjahr, 2000, 2003b].

The GBAS algorithm is based on the graph representation of a feasible solution to a Combinatorial Optimisation Problem as a walk in a directed graph called a *construction graph*.

Definition 5.2.1 *Let an instance of a Combinatorial Optimisation Problem be given. A* construction graph *is defined as a directed graph* $C = (V, A)$ *together with a function* ϕ *with the following conditions:*

1. *In C, a unique vertex is marked as the* start vertex*,*
2. *let W be the set of directed walks w in C satisfying the following conditions:*
   (a) *w starts at the start vertex of C*
   (b) *w contains each vertex of C at most once*
   (c) *The last vertex on w has no successor in C that is not already contained in w (that is, w cannot be extended without violating the previous constraint).*

Then ϕ maps the set W onto the set S of solutions of the given problem instance containing all feasible solutions. In other words: To each walk satisfying the constraints listed above, there corresponds, via ϕ, a solution in S, and to each solution in S, in particular to each feasible solution, there corresponds, via ϕ^{-1} at least one walk satisfying the listed constraints.

Including the construction graph GBAS consists of four other components:

- a set of agents,
- a set of transition probabilities,

- a set of trail values,
- and a set of attractiveness values.

The set of *agents M* is defined consisting of m agents, $M_1, \ldots, M_m$. Each agent performs a random walk with certain transition probabilities, building up a path through the construction graph. Agents may compute their paths either in parallel or sequentially, according to the implementation. Each time period that consists of all m agents completing a path is known as a *cycle*. An application of the algorithm consists of multiple cycles from $1, \ldots, max_{cycle}$. Although max_{cycle} is normally fixed in advance this is not obligatory and it is conceivable that an implementation could alter it during execution. In this thesis max_{cycle} is used interchangeably with max_{it}, where one cycle is the same as one iteration.

The second additional component is the set of *transition probabilities* $p_{kl}(n,u)$ that determine the distribution of moves at each stage of the construction process. The general equation of the probability distribution from a node k to a node l, given a partial path u in cycle n is given in Equation 5.1.

$$p_{kl}(n,u) = \begin{cases} \frac{[\tau_{kl}(n)]^{\alpha} \cdot [\eta_{kl}(u)]^{\beta}}{\sum_{r \notin u,(k,r) \in A} [\tau_{kr}(n)]^{\alpha} \cdot [\eta_{kr}(u)]^{\beta}} & l \notin u, (k,l) \in A \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

α and β are both defined to be greater than or equal to zero and are parameters to be supplied by the researcher. At the start of each cycle each agent is placed at the start node and for each step t the agent takes another random move. The cycle ends when there are no moves left to be made or when all transition probabilities are zero, at which point the path will correspond to a solution to the problem.

In Equation 5.1 values of $\tau_{kl}(n)$ are used. These values are the *trail intensities* and are defined to be greater than or equal to zero for each arc in the construction graph, $\tau_{kl}(n) \geq 0, \forall (k,l) \in A$. The values may differ according to each cycle and therefore are given the parameter n to indicate this variable nature. At the beginning of the first cycle each arc has the trail level of $\frac{1}{|A|}$ and at the end of each cycle an update is made using Equation 5.2. The solutions created by the m agents have objective values $f_1, \ldots, f_m$ and each $i \in \{1, \ldots, m\}$.

$$\Delta\tau_{kl}^{i} = \begin{cases} \phi_i(f_1,\ldots,f_m), & \text{if agent } M_i \text{ has traversed arc } (k,l) \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

ϕ_i is a non-negative reward function which is non-increasing in the corresponding variable f_i and may depend on the paths of the agents in the cycles $1,\ldots,n-1$. Let

$$C = \sum_{(k,l)\in A} \sum_{i=1}^{m} \Delta\tau_{kl}^{i} \tag{5.3}$$

If $C = 0$, then no change is made to $\tau_{kl}(n)$ for all arcs, as shown in Equation 5.4.

$$\tau_{kl}(n+1) = \tau_{kl}(n) \tag{5.4}$$

However if $C > 0$, then $\tau_{kl}(n)$ is updated using Equation 5.5.

$$\begin{aligned} &\tau_{kl}(n+1) = (1-\rho)\tau_{kl}(n) + \rho\Delta\tau_{kl}, \\ &\text{where } \Delta\tau_{kl} = \tfrac{1}{C}\textstyle\sum_{i=1}^{m} \Delta\tau_{kl}^{i} \end{aligned} \tag{5.5}$$

As in previous implementations ρ is called the *decay rate*. The equations are normalised so that $\sum_{(k,l)\in A} \tau_{kl}(n) = 1$, removing the requirement for various pheromone limits.

As before by setting ρ to zero the influence of the pheromone matrix is removed and therefore the random search becomes influenced only by the heuristic, which by setting $\beta = 0$ can also be eliminated.

The final component of the algorithm is the *attractiveness values*. This is the information that is supplied by the heuristic. The heuristic may depend on previously traversed paths or may be dynamically altered. The heuristic can be used not only to rate moves but also to block off infeasible solutions by returning a value of zero.

For the convergence proof to work a number of conditions are also required to be met by the implementation.

1. $\alpha > 0$.
2. There must be at least one *reachable* optimal walk.
3. Only walks that are at least as good as the best found walk, f^*, so far get a positive increment $\Delta\tau_{kl}^{i}$. At the start $f^* = \infty$.

These are common conditions met by many of the algorithms previously described in Chapter 2 and therefore should not be a surprise. The final condition does not have to be invoked immediately only after some point n_0 allowing a possible period of exploration without aggressive exploitation.

A design choice was made at this stage to make $n_0 = 0$. This means that initially all the solutions are added to the pheromone matrix and the global best is then recalculated to the $\min(f_1, \ldots, f_m)$. The reason why the pheromone matrix is updated with all the solutions of the first cycle is because the rule states to add all solutions whose solutions are less than f^*, which at the start is equal to infinity. This choice was made for the same reason as there is no q_0 parameter in AS and MMAS. By increasing n_0 it would increase the diversity of solutions being stored in the pheromone matrix setting up a bias to be exploited, this diversity can be accommodated for by setting other parameters less aggressively.

Another reason to remove both q_0 and n_0 is that there is no clear way to give equal levels of diversity to each algorithm via these parameters, therefore by removing them it gives each algorithm the same conditions and helps to standardise the experiments. This decision is critical and therefore forms a separate assumption underpinning the work in this chapter.

Assumption: The parameters n_0 and q_0 can be set to 0 and compensated for by less aggressive settings of the other algorithm parameters.

The pseudo-code for the GBAS algorithm is given in Figure 5.1, and Table 5.1 provides a summary of the differences between the three algorithms. For more information on Ant System and the Max-Min Ant System see Sections 2.2.1 and 2.2.6 respectively.

Figure 5.1: Pseudo-code for the Graph-based Ant System

```
1  Initialise pheromone trails τ_kl on the arcs (k,l) of the construction graph C;
2  for iteration n = 1,...,max_cycle do
3      for agent i = 1,...,m do
4          k ← current position of the agent, equal to the start node of C;
5          u ← ∅, current path of the agent;
6          while a feasible continuation (k,l) of the path u of the ant exists do
7              select successor node l with probability p_kl;
8              u ← u ∩ (k,l);
9          endw
10     endfor
11     do pheromone update;
12 endfor
```

Property	**GBAS**	**AS**	**MMAS**
Initial Pheromone Value (τ_0)	$\frac{1}{n^2}$	5	3
Minimum Pheromone Value (τ_{min})	0	0	$\tau_{min}(t) = \frac{\tau_{max} \cdot (1-\rho^{dec})}{avg \cdot p^{dec}}$ where $p^{dec} = \sqrt[n-1]{p_{best}}$ and *avg* is the average number of options the ant has to choose at any decision point
Maximum Pheromone Value (τ_{max})	1	None	$\tau_{max}(t) = \frac{1}{1-\rho} \cdot \frac{1}{L_{best}}$

GBAS	Pheromone Update Rule
$\tau_{ij}(t+1) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}$, where $\Delta\tau_{ij}(t,t+1) \leftarrow \frac{1}{C}\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$	
AS	Pheromone Update Rule
$\tau_{ij}(t+1) \leftarrow \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t,t+1)$ where $\Delta\tau_{ij}(t,t+1) \leftarrow \sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t,t+1)$	
MMAS	Pheromone Update Rule
$\tau_{ij}(t) \leftarrow \rho \cdot \tau_{ij}(t-1) + \Delta\tau_{ij}^{best}$	

Table 5.1: Table showing the explicit differences between GBAS, AS and MMAS.

5.3 Experimental Methodology

As this chapter deals with empirical experiments it is important to consider four issues when setting particular parameters and controlling variables in a suitable way.

- First, one has to consider the significance of the experiments performed. This is primary as without it at the centre of the experiments one could not make any useful claims. As part of this decision, the time to collect the results also has to be factored, creating a trade-off between significance and time taken.
- Second, the statistical tools are going to be used on the data. This requires consideration about what data is required to be captured from the particular experiment.
- The quality of the runs should be taken into account. Each algorithm should have fair and unbiased conditions for creating solutions of a quality expected in normal research.
- Finally, the running time of all the experiments should be taken into consideration. This is important as there are thousands of individual runs that must be performed and if one were to test every combination for an hour, the experiments would be comprehensive but would never complete.

Therefore with these four considerations in mind the parameters are being controlled in a deterministic manner to maximise the quality of the results.

All the following experiments made this assumption:

Assumption: The parameters involved in the probability rule are independent of those used in the trail update rule.

This assumption was postulated to cut down the parameter space and via exploratory runs was found to be consistent. In Table 5.2 the space of parameters has been illustrated. This shows which variables are considered independent and which are not.

A second assumption has been made for those experiments involving $\beta > 0$, thus introducing heuristic information into the algorithm, which is the following:

Assumption: In general, the values returned by the heuristic are valid indicators of good solutions.

The consequence of this assumption is that the heuristic will be treated like an oracle relaying information that will, in general, lead us to better solutions. Given the results

	α	β	ρ	m
α	NA	Yes	Independent	Independent
β	Yes	NA	Independent	Independent
ρ	Independent	Independent	Yes	Yes
m	Independent	Independent	Yes	Yes

Table 5.2: Experimental Parameter Space (NA=Not Applicable)

in Section 3.3 this is not a large assumption to make, but it does need to be made for clarification. In the next chapter this assumption will be discarded and misinformation returned in its place.

The experiments conducted with the parameter m were split into two groups. The first a direct setting of m from the range $[1,2n]$, where n is the number of components in the problem, and was allowed to run for 500 iterations. The second group of experiments varied, according to Equation 5.6, the maximum number of iterations that the algorithm could perform. This fixed the total number of samples equal to 1000, the idea being to test how effective the algorithms were at using a fixed number of samples. This group of experiments will be referred to as m_2.

$$max_{it} = \frac{1000}{m} \tag{5.6}$$

Initially experiments were run with all the TSPLIB problems with $n <= 100$. This gave an exploratory guide for the hypotheses at the end of the paragraph, which were then tested with a set of problems drawn from the TSP, QAP and TS libraries. The reason why the exploratory experimentation was done only with TSPLIB was not only one of simplicity, but the primary factor was that it enabled the other two domains to be used without any further bias towards favourable problems.

Each experiment is split into two parts one for GBAS and AS, the other for GBAS and MMAS. This eliminates the requirement for more sophisticated statistical reasoning where three or more variables are concerned. Once the correlations are calculated the decision to accept or reject the Alternative Hypothesis agrees with what the majority of the problems result in. No further decision test was done by grouping these correlations in some way as this test would not have any greater statistical significance. In general if the decision is not distinct there will be further discussion.

1. (a) GBAS and AS are significantly correlated in terms of variation in ρ,

(b) GBAS and MMAS are significantly correlated in terms of variation in ρ,

2. (a) GBAS and AS are significantly correlated in terms of variation in α and β,
   (b) GBAS and MMAS are significantly correlated in terms of variation in α and β,

3. (a) GBAS and AS are significantly correlated in terms of variation in m,
   (b) GBAS and MMAS are significantly correlated in terms of variation in m.

Each run in an experiment was repeated 25 times with the following problem sets (n corresponds to the number of cities for the TSP, the number of flows and locations in the QAP, and the number of shooting days in TS):

- 5 TSP problems were selected from preliminary experiments to demonstrate effectively how performance changes with n

  gr17, bayg29, brazil58, eil76, kroA100

- 8 QAP problems were selected, two problems at random were taken from each subgroup (see Sub-section 3.3.2)

UIGD	sko81, had18
URGI	tai50a, lipa80a
RLI	bur26f, esc16c
RLLI	tai100b, tai15b

- 5 TS problems, all from the same set of problems used for Chapter 4. One problem was chosen for each n at random. For these problems the optima are unknown and therefore the P_{GB} will be replaced with the global best (GB).

  talent_10_10_5, talent_10_15_8, talent_10_20_5, talent_10_30_4, talent_10_100_8

This led to a total number of 450 (18 problems * 25 trials) sets of experiments over a range of 3 domains and over values of n in the range $[1, 100]$.

The data variables collected were:

- P_{GB} - percentage distance from global optimum calculated as in Equation 5.7 (also referred to as the performance),

- I_{GB} - index of the last iteration resulting in a solution with a better objective value,

- T_{GB} - time taken to produce the solution closest to the global optimum,

where GB abbreviates Global Best.

$$\frac{\text{global best} - \text{known optimal}}{\text{known optimal}}\% \tag{5.7}$$

The first variable characterises the performance of the algorithm in terms of its ability to reach known optima. Where a known optimum is not known the actual value of the best solution will be used. The advantage of the percentage is that it makes it easier to compare performance over multiple problems.

The second variable gives an idea of the convergence of the particular algorithm. Convergence itself was not measured for each algorithm as there are multiple definitions as what constitutes convergence for a particular pheromone matrix and as these experiments are trying to deal with the practical implications, this measure was chosen as a valid substitute. Therefore in this chapter when convergence is mentioned this is what is meant.

Finally, time was included for completeness, but no results will be given as the times correlate with I_{GB}.

These variables were compared for correlation by using the Pearson Product-Moment Correlation on the medians of the 25 samples. This correlation was favoured over the Spearman Rank Correlation as there was enough sampling to satisfy the parametric nature of the test and the data was of a interval or ratio nature. It was also the case that the experiments were designed to make sure they lent themselves to a related paired test satisfying the final condition for the Pearson Product-Moment Correlation. In the tables displaying the results of these tests the values of r, which is the correlation coefficient, are given to two decimal places and the p-values, indicating the significance of the correlation, are given to four.

In each section examples of the graphs produced for each problem will be given. There are obviously more graphs than those shown but the ones chosen illustrate a particular anomaly or were representative of the others. Where the results differed between problem domains figures are included from each. This results in a large number of graphs which for the reason of better presentation are given with any tables at the end of each section.

The figures plot the non-parametric statistics of the results. Therefore the points are set at the median of the samples and the error bars setting out the interquartile range. The reasons for using these non-parametric statistics were given in Chapter 3. Finally,

any results that were derived from the original data and broken down to smaller groups will be given as integer percentages. This is because the precision of these figures is no longer significant due to the number of problems chosen for each domain but they indicate certain trends that are of note.

5.4 Variation in ρ-Values

In this section the decay rate of the pheromone matrix will be varied to see how the algorithms behave. For this ρ_{alg} will be used to represent the proportion of the pheromone matrix from the previous iteration (τ_{t-1}) used in a particular algorithm. This variable will take a value in the range of $(0,1)$, and for GBAS is a substitution for $(1-\rho)$ and for AS and MMAS it is equal to ρ.

In exploratory experimentation it was found that the graph of performance against ρ_{alg} was not a linear relationship. As the proportion of the pheromone matrix tended to 1, especially for larger problems, the performance would degrade. Therefore the experiment was split into two areas. The first tested values in the range $\rho_{alg} \in (0, 0.9]$, while the second tested values in the range $[0.9, 1.0)$. The boundary of 0.9 was chosen as in the majority of cases this was where the performance change in AS and MMAS started to significantly differ from that of GBAS. In the case of GBAS the performance would continue to improve until some later unknown value of ρ_{alg}, while for AS and MMAS the performance would turn upward indicating a degradation in performance.

It is worth pointing out that problems with equal values of n may have different optimal ρ_{alg} values. This is best illustrated for GBAS in Figure 5.2 where as ρ_{alg} tends to 1 gr48 and att48 continue to improve their performances, whereas hk48 displays a slight up-turn in the trend line. In this particular case the difference is only significant (Mann-Whitney U-test, p-values < 0.05) for the final two points at $\rho_{alg} \in \{0.95, 0.99\}$ but this may increase on more extreme problems. Unfortunately there was not time to do further research into this but it may yield interesting results for how the algorithm is responding to various traits in the landscape of the individual problem.

For the following experiments the parameters were set as listed below.

- $\alpha = 1$,
- $\beta = 0$,

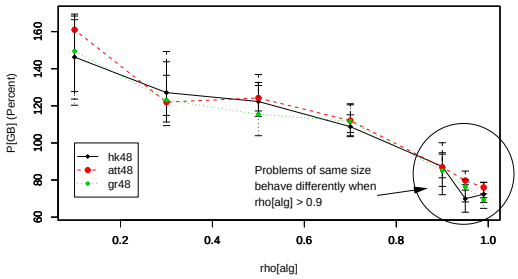


Figure 5.2: Graph illustrating how problems with the same number of cities can have different performances depending on individual landscapes. The example is of GBAS applied to hk48,att48 and gr48 from TSPLIB.

- $m = n$,
- $\rho_{alg}^{high} \in \{0.9999, 0.9995, 0.999, 0.995, 0.99, 0.95, 0.9\}$,
- $\rho_{alg}^{low} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

β has been set to zero to detect the influence of ρ on the pheromone matrix. Using a heuristic would alter the performance, thus removing this parameter reduced the complexity of the algorithms. m is set equal to n to give the algorithms a chance to maximise their performance given the other experimental conditions.

The eight hypotheses for this experiment have the following pattern, where R is either ρ_{alg}^{low} or ρ_{alg}^{high}:

Null Hypothesis:	There is no significant correlation between GBAS and AS when varying ρ_{alg} in the range R for the variable P_{GB}.
Alternative Hypothesis:	There is a significant correlation between GBAS and AS when varying ρ_{alg} in the range R for the variable P_{GB}.

The same hypothesis is being tested for MMAS in place of Ant System (AS). These hypotheses divide the requirements into two areas, performance in terms of percentage from known optima (P_{GB}) and iteration of last improvement (I_{GB}).

Figure 5.3 illustrates the results for the TSP problem eil76. For ρ_{alg}^{low}, the general trend is for MMAS to have the steepest improvement, eventually converging after the

recorded period. This would suggest that the additional complexity of the pheromone matrix for this algorithm is well suited to this problem and may be related to the convexity of the search space.

The results of the runs were then correlated using the Pearson Product-Moment Correlation to test how the algorithms compared to GBAS. For the TSP, both the performance of AS and MMAS are significantly correlated, as indicated in Table 5.3. This is not to say they achieve similar performance, but merely that one can expect the performance to move in the same direction with the same change in ρ_{alg}. For I_{GB} in Table 5.4, Ant System is significantly correlated to GBAS but MMAS is not, concluding that the dynamic updates to τ_{min}, τ_{max} enable the algorithm to delay its convergence.

When higher values in the range $[0.9, 1.0)$ are used for ρ_{alg}, both algorithms show no significant degrees of correlation with GBAS for P_{GB} or I_{GB}. The reasons for this are not clear but most probably depend on the update rule and the normalisation process that GBAS undertakes. Preliminary experiments were carried out to see if the correlation between AS and GBAS could be improved by altering the update rule from Equation 5.8, as used in AS, to Equation 5.9, as used in GBAS.

$$\tau_{ij}^{(t)} = \rho \cdot \tau_{ij}^{t-1} + \frac{Q}{f_{opt}} \tag{5.8}$$

$$\tau_{ij}^{(t)} = (1-\rho) \cdot \tau_{ij}^{t-1} + \rho \cdot \frac{Q}{f_{opt}}, \text{ known as the } AS_{weighted-\rho} \tag{5.9}$$

The experiments were then run to test the following hypotheses:

Null Hypothesis:	There is no difference between AS_ρ and $AS_{weighted-\rho}$.
Alternative Hypothesis:	There is a difference between the P_{GB} values between AS_ρ and $AS_{weighted-\rho}$.

The runs were performed on all TSPLIB problems with $n <= 100$ and the p-values of the hypothesis testing were all greater than 0.05. These results showed that the two rules were identical, leaving only the differences in the other parts of the update rule.

In regard to the QAP problems, for ρ_{alg} values in the range $(0, 0.9]$, the performance correlations are significant on most of the problems for both MMAS and AS. Furthermore, Ant System is highly correlated with GBAS in its convergence characteristics in this range. MMAS takes twice as long to converge as the other two algorithms, which

converge within approximately 200 iterations. With this comes a clear advantage and MMAS demonstrates this by achieving better solutions than the other two in Figure 5.3. In contrast with Ant System, MMAS does not show significant correlation with GBAS.

When the ρ_{alg} values are raised above 0.9 GBAS's performance improves due to an increase in the number of iterations it takes to convergence. At this stage the algorithm shows a four-fold increase in the number of iterations it takes to converge, showing that it favours ρ_{alg} values higher than both AS and MMAS. In this range, neither of these algorithms show correlations to GBAS.

For the Talent Scheduling problems a similar picture emerges. For ρ_{alg}^{low} both Ant System and MMAS correlate significantly with GBAS, but when $\rho_{alg} \rightarrow 1$, as with the other two problem sets, there is no correlation. The convergence of Ant System correlates significantly with GBAS in the lower range but as ρ_{alg} increases the correlation is no longer significant. MMAS does not correlate to GBAS in either range for I_{GB}.

For the TSP problems, the Ant System is the worst performing algorithm in both ranges of values, while MMAS achieves the best solutions. In terms of the number of iterations taken to achieve their best solutions, MMAS takes the longest, thus it is unsurprising that it finds the best quality solutions. As ρ_{alg} increases above 0.9, GBAS improves, achieving better solutions and managing to delay convergence as late as MMAS.

Figures 5.4 and 5.5 depict both the QAP and Talent Scheduling problems respectively. In both of these figures Ant System performs better than GBAS but they follow similar paths. MMAS takes the lead in both ρ_{alg} ranges demonstrating that the ability to delay convergence is beneficial.

In all the graphs, after $\rho_{alg} = 0.99$, the impact on the solution quality is negligible. This is because so much of the previous pheromone matrix is kept that the distributions change very little over time.

The conclusions drawn from the figures and tables of correlations about the hypotheses stated at the start of this section are as follows:

- For ρ_{alg}^{low} and P_{GB}, GBAS and AS are significantly correlated, the Null Hypothesis is rejected.
- For ρ_{alg}^{low} and I_{GB}, GBAS and AS are significantly correlated so the Null Hypothesis

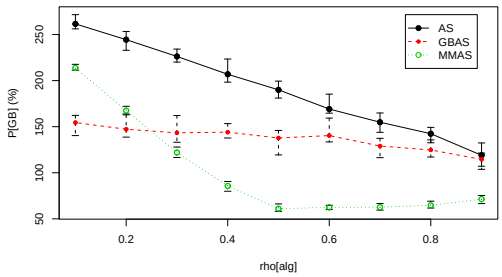
is rejected.

- For ρ_{alg}^{low} and P_{GB}, GBAS and MMAS are significantly correlated so the Null Hypothesis is rejected.
- For ρ_{alg}^{low} and I_{GB}, GBAS and MMAS are not significantly correlated so the Alternative Hypothesis is rejected.
- For ρ_{alg}^{high} and P_{GB}, GBAS and AS are not significantly correlated so the Alternative Hypothesis is rejected.
- For ρ_{alg}^{high} and I_{GB}, GBAS and AS are not significantly correlated so the Alternative Hypothesis is rejected.
- For ρ_{alg}^{high} and P_{GB}, GBAS and MMAS are not significantly correlated so the Alternative Hypothesis is rejected.
- For ρ_{alg}^{high} and I_{GB}, GBAS and MMAS are not significantly correlated so the Alternative Hypothesis is rejected.

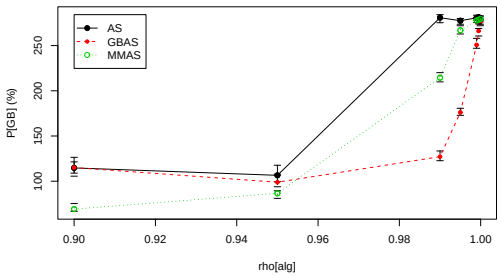
To conclude, the picture is more complicated than was expected. GBAS is a good model for how varying ρ_{alg} affects the algorithms AS and MMAS for values in the range $(0, 0.9]$. However for higher values the behaviour is algorithm dependent, indicating that the various pheromone update models play a significant role in the convergence and performance of the algorithms.

The delayed convergence of MMAS is most likely due to the dynamic nature of the limits on the pheromone matrix. Lack of significant correlations with GBAS for both AS and MMAS in ρ_{alg}^{high} is most likely due to the normalisation that occurs in GBAS to make the pheromone matrix sum to one. Confirmation of the effect of normalisation of the pheromone matrix on the behaviour of the Ant algorithm remains a subject for a later study.

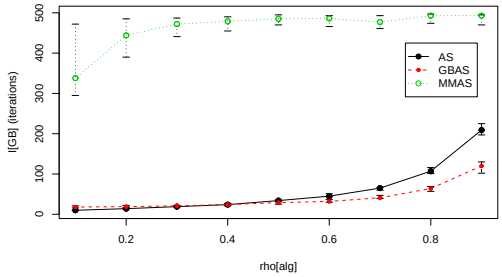
A *typical* value of ρ_{alg} is between 0.7 and 0.95. This is not damaging to the individual algorithm. However, if the particular experiment has any assumptions based on the performance of another algorithm, these assumptions are more likely to be invalid the higher ρ_{alg} is set.



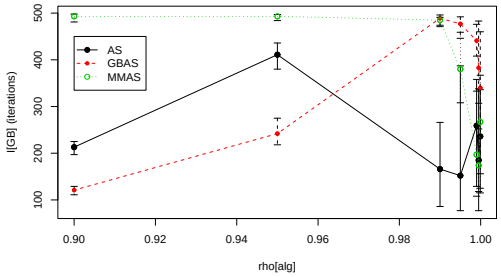
(a) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$



(b) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$



(c) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$

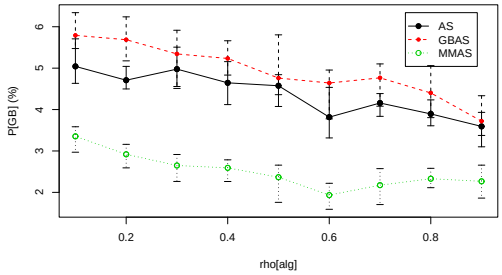


(d) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$

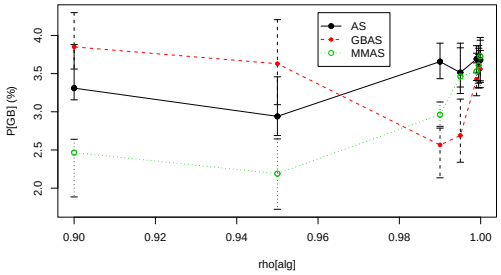
Figure 5.3: Example graphs of the performance of GBAS, AS and MMAS on a TSP problem, in this case eil76, while varying ρ_{alg}. Figures (a) and (c) on the left-hand side are in the range $(0, 0.9]$ while (b) and (d) are in the range $[0.9, 1.0)$.

Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
$\rho \in (0, 0.9]$			
TSP	5	5	5
QAP	8	8	6
TS	5	4	4
$[0.9, 1)$			
TSP	5	1	5
QAP	8	1	0
TS	5	0	0

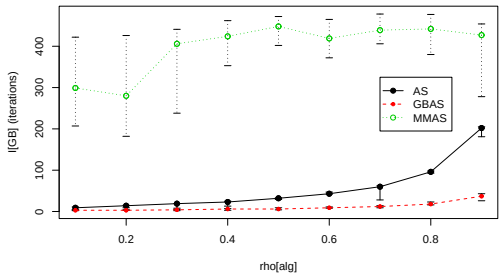
Table 5.3: Table summarising the hypothesis test results for P_{GB} varying ρ_{alg}. (For more detailed results refer to Table B.1.)



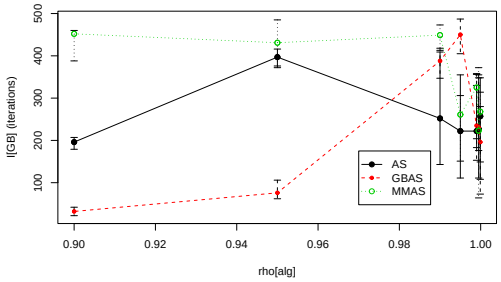
(a) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$



(b) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$



(c) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$

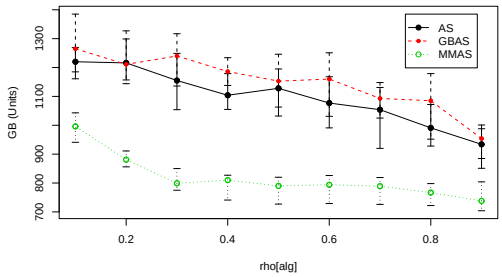


(d) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$

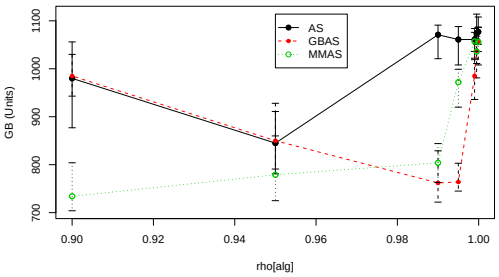
Figure 5.4: Example graphs of the performance of GBAS, AS and MMAS on a QAP problem, in this case bur26f, while varying ρ_{alg}. Figures (a) and (c) on the left-hand side are in the range $(0, 0.9]$ while (b) and (d) are in the range $[0.9, 1.0)$.

Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
$\rho \in (0, 0.9]$			
TSP	5	5	0
QAP	8	8	0
TS	5	4	0
$[0.9, 1)$			
TSP	5	0	0
QAP	8	0	0
TS	5	0	0

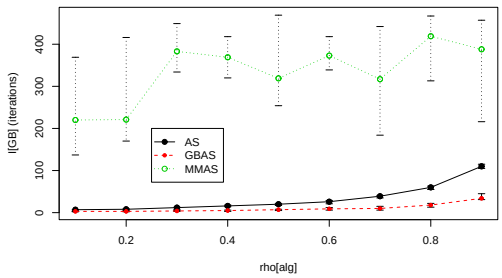
Table 5.4: Table summarising the hypothesis test results for I_{GB} varying ρ_{alg}. (For more detailed results refer to Table B.2.)



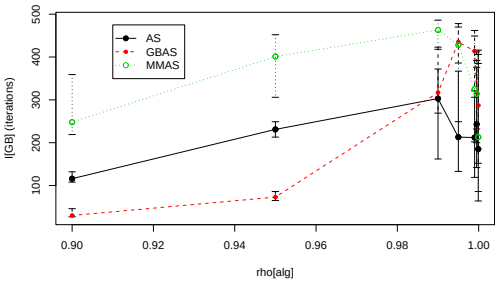
(a) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$



(b) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$



(c) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$



(d) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$

Figure 5.5: Example graphs of the performance of GBAS, AS and MMAS on a TS problem, in this case talent_10_20_5, while varying ρ_{alg}. Figures (a) and (c) on the left-hand side are in the range $(0, 0.9]$ while (b) and (d) are in the range $[0.9, 1.0)$.

5.5 Variation in α,β-Values

The parameters α and β are used to control how the relevant information sources are weighted together when deciding which node to visit next in the construction graph. α controls the weighting of the pheromone intensity (τ), while β controls the weighting of the heuristic (η). When both these are zero, the search is the standard random search, as each is then increased with respect to the other the validity of the values used takes on greater significance.

The objective of this experiment is to see how manipulating these two variables alters the performance of the particular algorithms, and to see if there are correlations between GBAS and the two other implementations. There are two trends that will expected to be observed, which are as follows:

- As $\alpha \to 5$ performance will at first increase, but then decrease as the number of iterations for the algorithm to converge is reduced. This reduction will be due to overemphasis on early choices.
- As $\beta \to 5$ performance will increase, but at the cost of faster convergence, which for more complicated landscapes may be a problem. This premature convergence will be due to overemphasis on the heuristic.

It is expected that the three algorithms will be significantly correlated as they use the same probability rule as in Equation 5.10, but it may be that the convergence of the algorithms differ in some way.

$$p(i,j) \propto \tau_{ij}^{\alpha}(t) \cdot \eta_{ij}^{\beta} \tag{5.10}$$

The hypotheses tested in this section follow the pattern below. In total there are twenty hypotheses with β replacing α, I_{GB} replacing P_{GB}, MMAS replacing AS, and separate hypotheses for each domain.

Null Hypothesis:	There is no significant correlation between GBAS and AS when varying α in the range $[0,5]$ for P_{GB} for TSPs.
Alternative Hypothesis:	There is a significant correlation between GBAS and AS when varying α in the range $[0,5]$ for P_{GB} for TSPs.

To test how the three algorithms perform with relation to these two parameters, the following conditions were set:

- $m = n$, where m is the number of ants and n is the size of the problem,
- $\rho_{alg} = 0.95$.

ρ_{alg} was set at 0.95 as this was the value a large number of the previous applications used, thus it was chosen to make the experiment applicable to previous work. This value is in the area where the algorithms do not show significant correlation; however the value of ρ should not affect the comparison as it is assumed to be independent of α and β.

α and β were then varied in the range $[0,5]$, creating 36 tests per problem, and with 25 runs per test set, this creates 9,000 individual experiments comprising the TSP and Talent Scheduling problems. For the QAP problems only α was varied as no heuristic is used for these, resulting in 1,200 experiments. In total this makes 10,200 tests.

In Figure 5.6 an example of the results from the TSP data set is given for varying α. One can observe for all three algorithms the performance graphs look very similar. For $\beta = 0$, they all show an initial improvement as α changes from 0 to 1, but then performance decays as the related I_{GB} results decrease quickly as α tends to five. The optimal combination for all algorithms is when $\alpha = 1, \beta >= 2$, although as β increases the algorithms are pushed to local optima more quickly.

In Figure 5.7, a similar set of graphs show the results for the TSP when β is varied. Again all three sets of graphs are similar with performance increasing as β tends to 5. Along with this increase in performance, when $\alpha = 0$ the convergence of the algorithms is delayed equally long for $\beta \in \{0,2,5\}$ which is contrary to expectation. This shows that an increase in β is not necessarily a step towards early convergence. This resilience is demonstrated throughout the graphs, therefore the early convergence must be a product of the reinforcement strategy and not of the heuristic itself.

Figures 5.8 and 5.9 are the analogous graphs for showing an example of the Talent Scheduling results. These are more entangled but similar trends are clear. Once again the values of 1 and 2 for α and β seem to be optimal. In the latter figure, as $\beta \to 5$ there is degradation in performance for all α indicating an optimal value for β. This is in contrast to Figure 5.7 for the TSP where as $\beta \to 5$ the performance continues to perform optimally without degradation.

Finally, the QAP problems are represented by Figure 5.10. A setting of 1 for α provides the best solutions for GBAS with convergence occurring very quickly when α gets above this. Ant System is more robust with its choice of α, because the choice does not seem to impact the convergence of the algorithm as dramatically. MMAS on the other hand follows, in a less extreme manner, the same decay of convergence as GBAS.

For AS and MMAS, when $\alpha = 1$ the best solution is found after $\alpha = 0$; it was expected the convergence would be more like the graph for GBAS. This observation shows that, for these settings, the algorithms have been able to increase the quality of their solutions for longer, indicating that the search is performing well. For GBAS and MMAS, there is more concern over the fall in convergence as α increases above 1. This demonstrates that the algorithm is quick to find new solutions, but then will find itself too often in a local optimum.

To find out how the two algorithms, AS and MMAS, compared to GBAS the results were compared in terms of rank. For each point in a graph, the rank of each α point with respect to other α values, is calculated in the GBAS results and then compared to the rank of the associated point in either the AS or MMAS results. In this manner the percentage of point ranks that the AS and MMAS had in common with GBAS was calculated. The results for when α was varied are shown in Table 5.6, and for β in 5.7.

The first two columns of each table refer to the variable P_{GB}. Table 5.6 shows that both TSP and Talent Scheduling have about 50% of points are in agreement with GBAS. For the β values in Table 5.7 the TSP proportion is almost double that of Talent Scheduling, for both Ant System and MMAS. In contrast to Table 5.6, the values for QAP and Talent Scheduling differ for each algorithm. Ant System seems to show less agreement than MMAS for these two domains with values in the range of 0-50%. In contrast, MMAS has values in the range 17-67%. The conclusions that can be drawn from these results are that MMAS behaves more like GBAS when α is varied than Ant System does, and that when varying β both AS and MMAS act like GBAS to a similar degree.

In both tables the results for I_{GB} are mid-range percentages, showing that there may be a reasonable correlation between the three algorithms. It seems that the results for the TSP may be more favourable than those of the Talent Scheduling especially for when β is varied. This may be caused by the quality of the heuristic and the strength of the update on the matrices as a result.

Finally it leaves the correlations to be described. In Tables B.3 to B.10 the results of

the Pearson Product-Moment Correlation calculations and their associated p-values are given. In some rows there is the abbreviation NA where the points were identical between the two algorithms' results, thus the correlation is 1. The significantly correlated p-values in the tables are marked with an asterisk. Summary tables are given in Table 5.8 and 5.9.

In Table 5.8 the performances of both the algorithms are significantly correlated for most of the TSP problems when α is varied. In contrast, the QAP and Talent Scheduling problems are not significantly correlated. This is probably due to the difference in the landscapes of the two domains. However, MMAS in most cases is correlated with GBAS for these domains, which is unexpected considering these are the most dissimilar algorithms. It is likely that there is something about these domains that enable MMAS and GBAS to behave in a similar way.

Table 5.8 shows both algorithms exhibiting significant correlations for nearly all the problems and α values. This is not surprising as the heuristic is the same for all the algorithms, therefore it should have the same effect on the probability rule. In the few cases where the correlations are not significant it would seem that the GBAS graphs are likely to show smoother trendlines. The most likely reason is that Ant System and MMAS have found a significantly better solution than GBAS did, thereby producing a greater change between β values.

In Table 5.9 the results for variable I_{GB} are summarised. The table reveals little significant correlation between the algorithms. For Ant System, a few significant correlations appear for small TSP and Talent Scheduling problems; while for MMAS, only the TSP shows any sizable correlations. This is probably because of how the update rules are manipulating the pheromone matrix values. For GBAS, the results for the variable I_{GB} display a smooth decay trend, which is perhaps a symptom of the normalisation factor; while the other two algorithms have a peak at $\alpha = 1$, which is most likely where the trends become out of synchronisation.

Table 5.5 shows which of the hypotheses set out at the start of the section have been rejected for each combination of parameters. In addition there are a number of caveats to these results that could alter the conclusions if given more evidence. The first applies to the Ant System algorithm, while varying α for the variable I_{GB} on the TSP and Talent Scheduling problems, where for small n, instead of rejecting the Alternative Hypothesis, it might be possible to reject the Null Hypothesis. The second observation

		AS		MMAS	
		α	β	α	β
TSP	P_{GB}	N	N	N	N
	I_{GB}	A	N	A	N
QAP	P_{GB}	A	–	A	–
	I_{GB}	A	–	A	–
TS	P_{GB}	A	N	N	N
	I_{GB}	A	A	A	A

Table 5.5: Table of which hypotheses have been rejected for Section 5.5. (N=Null Hypothesis rejected A=Alternative Hypothesis rejected)

is that, for the convergence of MMAS when $\beta \in \{2,3\}$, the results are significantly correlated for all problems. These two observations are given to show the complexity of the results and that more evidence could shift the conclusions significantly.

Therefore to complete this section the observations are summarised:

- $\beta \to 5$ P_{GB} improves but I_{GB} is significantly reduced.
- $\alpha = 1$ is the optimal setting combined with $\beta = 2$ if a heuristic exists.
- For both algorithms, there is little significant correlation with GBAS when varying α in terms of convergence.
- Both AS and MMAS correlate significantly with GBAS on the TSP problems for both variables and parameters. For Talent Scheduling the correlations show mixed results, and for QAP there is no correlation with GBAS for either algorithm. This demonstrates that the modelling of the parameters α and β is dependent on the domain and may differ significantly between algorithms.
- The role of β, and therefore heuristics, in achieving good solutions and on the convergence, of all three algorithms has been shown to be very important.

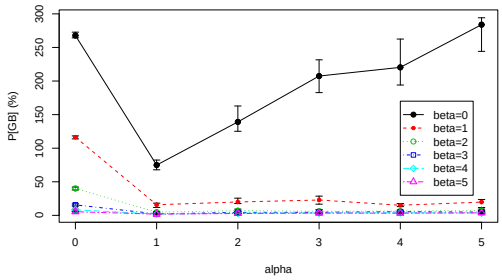
In general, it can be concluded that the pheromone matrix is a source of difference between the behaviours of the various algorithms, while the heuristics are a source of similarity. This may be an obvious statement, but it is important that this difference is taken into account when discussing models, such as GBAS with respect to different algorithms.

It is clear from these results that the heuristic plays a central role in guiding the algo-

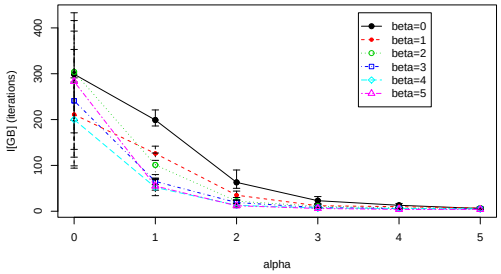
Problem	**PRO(AS)** P_{GB}	**PRO(MMAS)** P_{GB}	**PRO(AS)** I_{GB}	**PRO(MMAS)** I_{GB}
gr17	36	47	61	67
bayg29	81	75	81	72
brazil58	53	56	36	64
eil76	50	53	53	67
kroA100	47	44	42	64
had18	17	50	50	50
sko81	33	17	0	0
tai50a	33	67	17	17
lipa80a	33	50	17	17
esc16c	0	17	50	50
bur26f	33	67	17	17
tai15b	50	33	33	33
tai100b	17	50	0	0
talent_10_10_5	31	33	64	58
talent_10_15_8	28	50	83	61
talent_10_20_5	42	47	89	64
talent_10_30_4	47	56	61	53
talent_10_100_8	19	36	14	58

Table 5.6: Table showing how AS and MMAS rankings compare to GBAS for various α values.(PRO(X)=Percentage of points from the algorithm X that are in the same rank as in the corresponding GBAS runs)

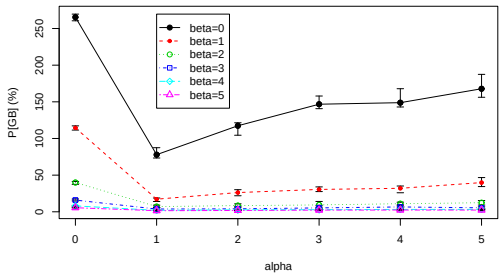
rithm and that a good heuristic applied strongly can produce accurate results. However, the cost of this is a reduced ability to search the landscape in areas that the heuristic does not provide productive values for. This emerges due to fast convergence of the algorithms.



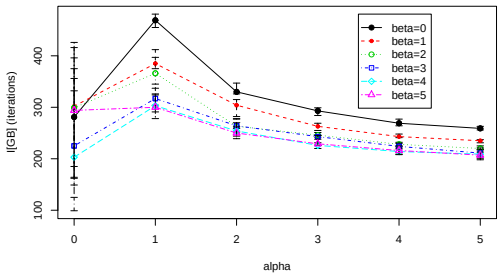
(a) P_{GB} using GBAS



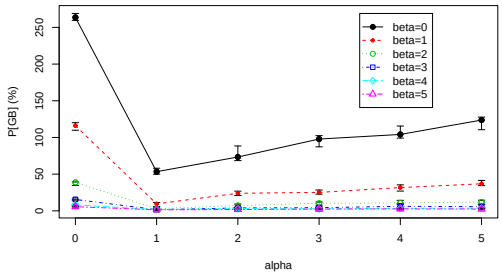
(b) I_{GB} using GBAS



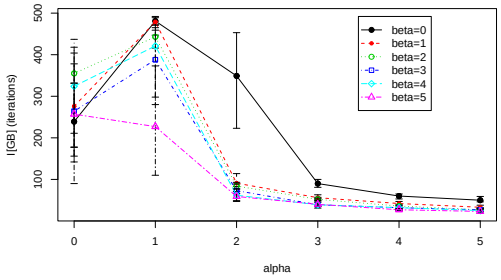
(c) P_{GB} using AS



(d) I_{GB} using AS

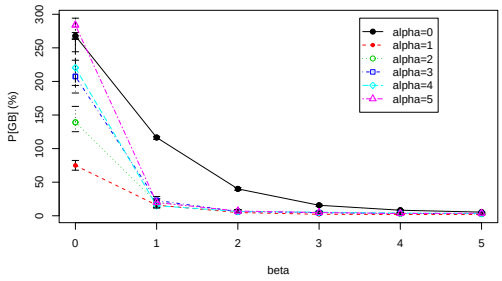


(e) P_{GB} using MMAS

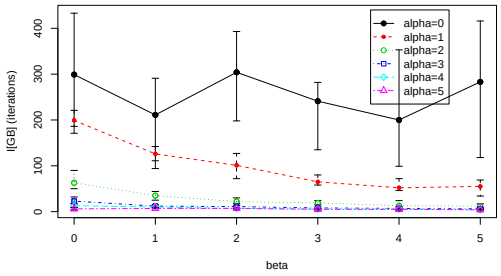


(f) I_{GB} using MMAS

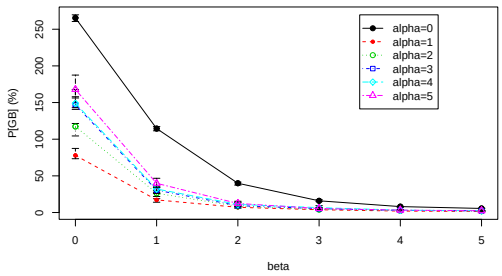
Figure 5.6: Example of the results achieved by varying α with GBAS, AS and MMAS on the TSP data set using Brazil58.



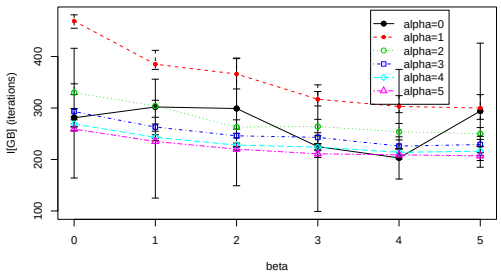
(a) P_{GB} using GBAS



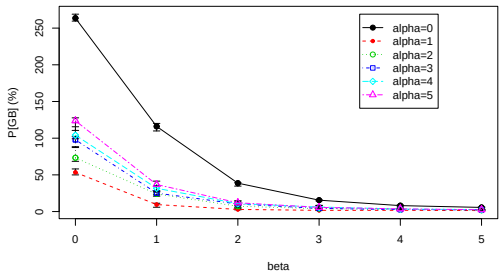
(b) I_{GB} using GBAS



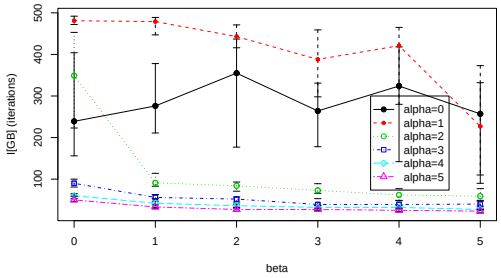
(c) P_{GB} using AS



(d) I_{GB} using AS

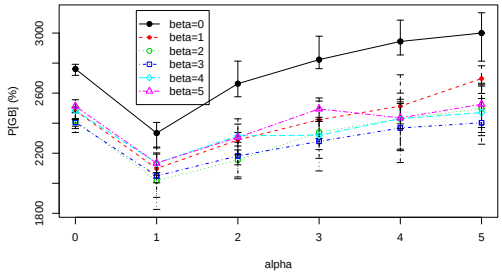


(e) P_{GB} using MMAS

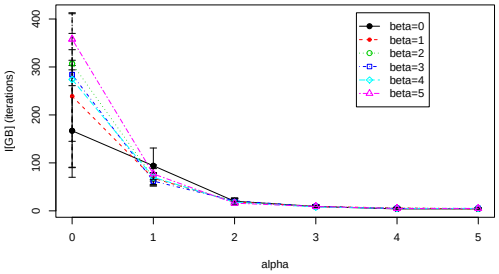


(f) I_{GB} using MMAS

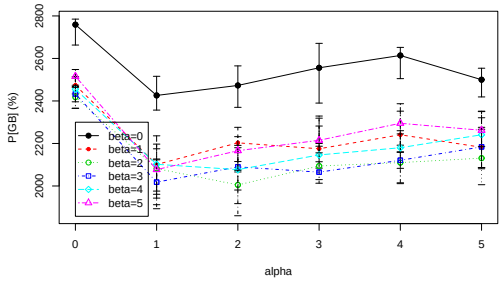
Figure 5.7: Example of the results achieved when varying β with GBAS, AS and MMAS on the TSP data set using Brazil58.



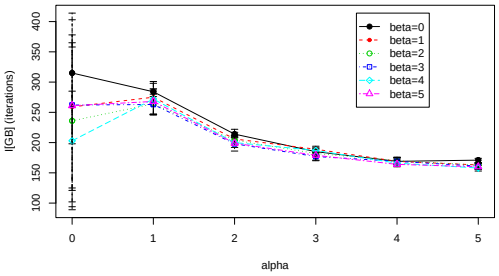
(a) P_{GB} using GBAS



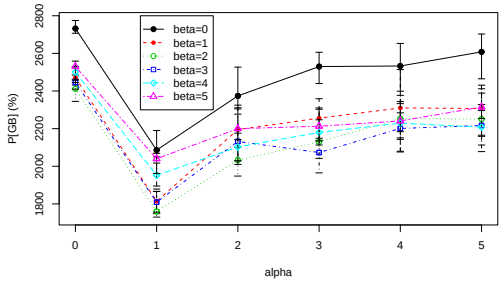
(b) I_{GB} using GBAS



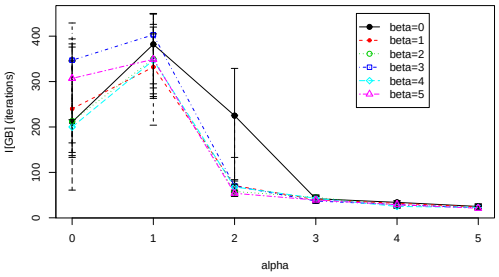
(c) P_{GB} using AS



(d) I_{GB} using AS

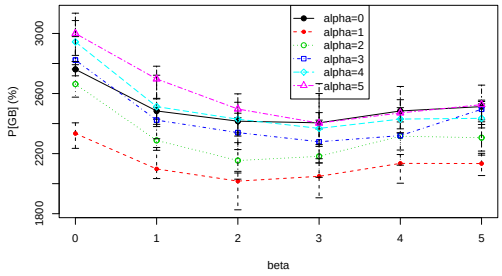


(e) P_{GB} using MMAS

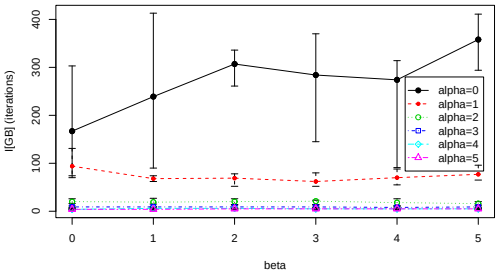


(f) I_{GB} using MMAS

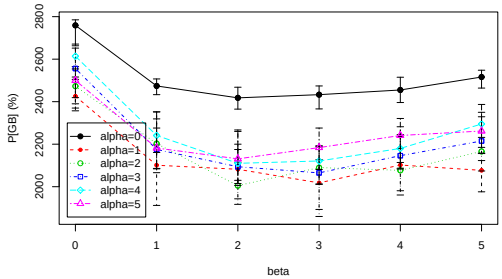
Figure 5.8: Example of the results achieved when varying α with GBAS, AS and MMAS on the TS data set using talent_10_30_4.



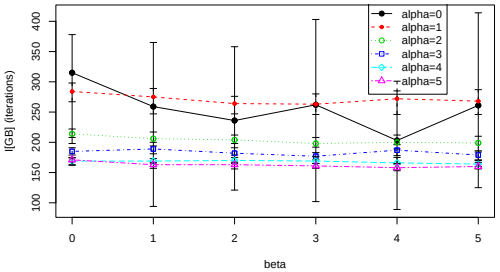
(a) P_{GB} using GBAS



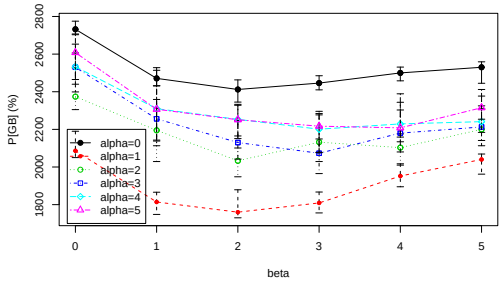
(b) I_{GB} using GBAS



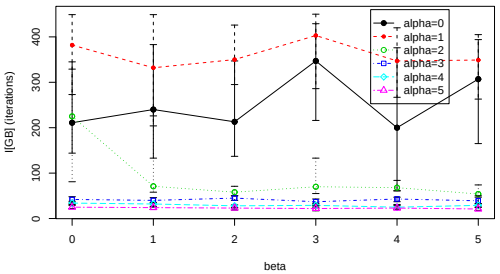
(c) P_{GB} using AS



(d) I_{GB} using AS

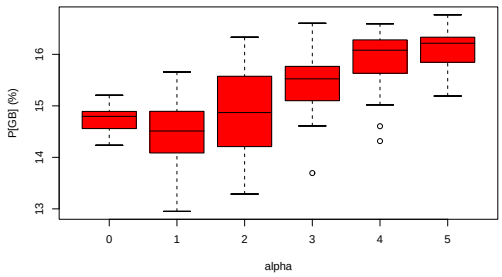


(e) P_{GB} using MMAS

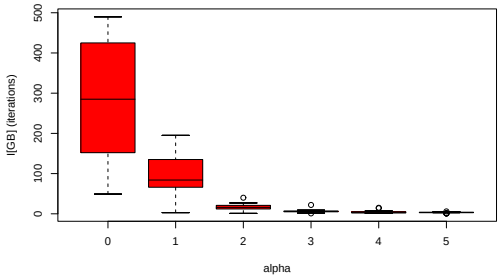


(f) I_{GB} using MMAS

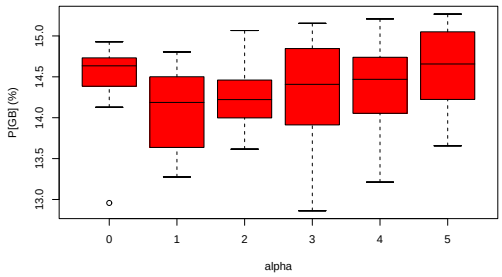
Figure 5.9: Exemplar of results varying β with GBAS, AS and MMAS on the TS data set using talent_10_30_4.



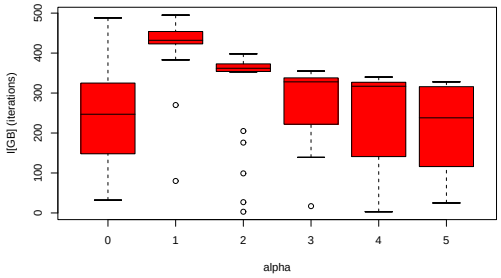
(a) P_{GB} using GBAS



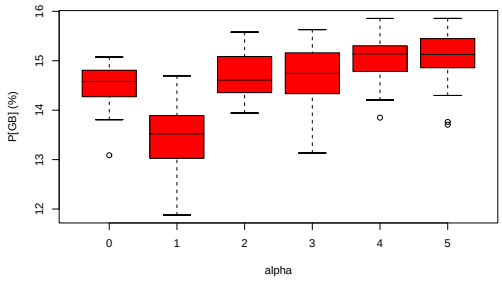
(b) I_{GB} using GBAS



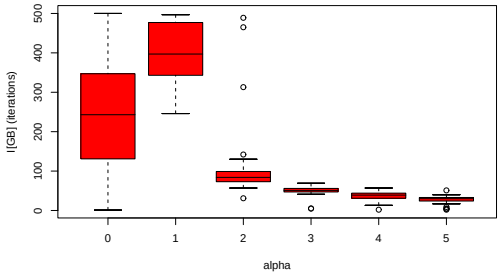
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 5.10: Example of the results achieved when varying α with GBAS, AS and MMAS on the QAP data set using tai50a.

Problem	**PRO(AS)** P_{GB}	**PRO(MMAS)** P_{GB}	**PRO(AS)** I_{GB}	**PRO(MMAS)** I_{GB}
gr17	72	61	42	44
bayg29	97	92	75	58
brazil58	94	89	56	64
eil76	83	83	67	58
kroA100	100	100	56	50
talent_10_10_5	47	19	31	28
talent_10_15_8	36	39	22	22
talent_10_20_5	44	36	33	17
talent_10_30_4	42	44	19	14
talent_10_100_8	53	50	36	19

Table 5.7: Table showing how AS and MMAS rankings compare to GBAS for various β values. (PRO(X)=Percentage of points from the algorithm X that are in the same rank as in the corresponding GBAS runs)

Varying Parameter	Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
α	TSP	30	23	23
	QAP	8	0	4
	TS	30	2	20
β	TSP	30	30	30
	QAP	8	NA	NA
	TS	30	28	21

Table 5.8: Table summarising the hypothesis test results for P_{GB} varying α and β. (For more detailed results refer to Tables B.3, B.4, B.5 and B.6.)

Varying Parameter	Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
α	TSP	30	6	8
	QAP	8	1	0
	TS	30	15	8
β	TSP	30	22	18
	QAP	8	NA	NA
	TS	30	3	2

Table 5.9: Table summarising the hypothesis test results for I_{GB} varying α and β. (For more detailed results refer to Tables B.7, B.8, B.9 and B.10.)

5.6 Variation in m-Values

The third experiment focuses on the number of ants that the algorithm creates per iteration (m). There is no theoretical way to predict what the optimal value of m should be a priori as it depends on an element of randomness and the complexity of the problem landscape. Therefore the question is how does the behaviour of the algorithms change when m is varied? Furthermore, is there a point at which creating more solutions is no longer of value?

In the majority of research one of two views is taken. The first is that m should be equal to the number of components in a solution; for instance in the TSP the value should be the number of cities. The second view is that very few ants are needed, and that the value is robust so that any number in the range 1 to 10 ants is adequate.

This section hopes to answer two questions, the first is to which view does GBAS subscribe, and the second is does the behaviour of both Ant System and MMAS correlate with GBAS or not? These aspects are summarised in the following hypothesis. This hypothesis can be replicated for MMAS and I_{GB}, producing four hypotheses in total.

Null Hypothesis:	There is no significant correlation between GBAS and AS when varying m for P_{GB}.
Alternative Hypothesis:	There is a significant correlation between GBAS and AS when varying m for P_{GB}.

To answer both these questions this section will be split into two experiments. Firstly, each run will consist of m ants per iteration, and it will last for a predetermined number of iterations. The second experiment will run each algorithm for 1000 ants. The latter tries to capture the *efficacy* of the sampling being undertaken by the algorithm. Only at the end of the second experiment in Section 5.7 will the questions posed above be able to be answered.

Varying the value of m is important to see how the algorithms perform with different amounts of sampling per update. The expectation is that the more ants that are used the better the quality of the solutions. The parameter is varied as follows:

- for $n <= 50$, $m \in \{1, \ldots, n\}$,
- for $n > 50$, $m \in \{1, \ldots, 0.25n, 0.5n, n, 2n\}$.

With the other parameters set as follows:

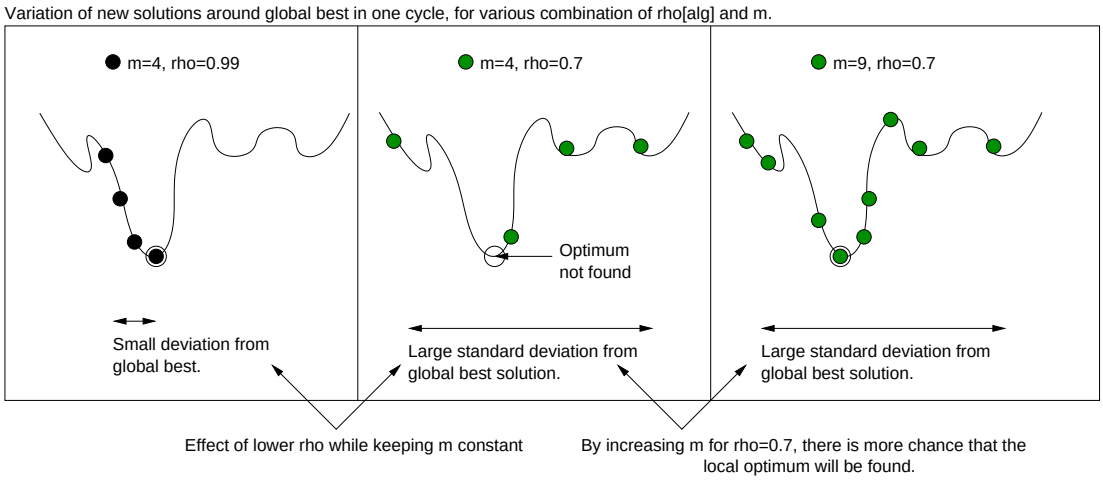


Figure 5.11: Figure illustrating why ρ_{alg} might vary with m.

- $\alpha = 1$,
- $\beta = 0$,
- $\rho_{alg} \in \{0.7, 0.9, 0.95, 0.99\}$.

ρ_{alg} was varied within a limited range, because m and ρ are assumed not to be independent. The variation was designed to show that large ρ_{alg} values would keep more information in the pheromone matrix, thus requiring less samples, and therefore a lower m, to move to a better solution. However, if m was large this would suit a lower ρ_{alg} value that would lose more of its information, thus requiring greater numbers of samples to maintain its search in the desired area. For instance, if one were to measure the standard deviation of solutions from the global best, to get the same number of solutions from the area around the global best when ρ_{alg} is set to a low value as when ρ_{alg} is set to a high value, the number of ants would have to be increased. Figure 5.11 gives a graphical representation of this.

Besides correlations, the results were viewed with respect to a set of four criteria:

- the ranking of ρ_{alg} for P_{GB} when $m = n$,
- the ranking of ρ_{alg} for I_{GB} when $m = n$,
- predictability of the rankings of ρ_{alg} as m varies,
- agreement of AS and MMAS rankings with the rankings of GBAS.

The first two criteria were to test the performance of each ρ_{alg} for this value of m to

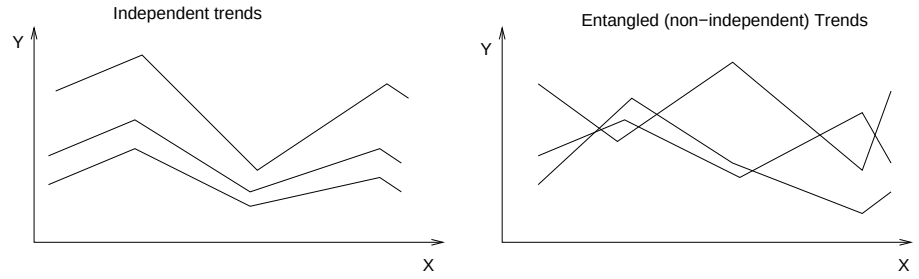


Figure 5.12: Figure showing independent and non-independent trends. These are the two cases the ranking of ρ_{alg} will identify.

set a standard ranking for the algorithm. The value of n was chosen as this is the most commonly quoted value for experiments to use. The third criteria is a measure of how well the ranks of ρ_{alg} values for P_{GB} and I_{GB} at $m = n$ hold for the other m values. These results will be given as a percentage of the total number of points tested as in Equation 5.11.

$$\begin{aligned} &\text{Prediction(algorithm)} = \frac{\sum_{m=1}^{m=2n} O(m)}{\text{number of points}} * \frac{100}{1} \\ &\text{where } O(i) \begin{cases} 1 & \text{if ranks are the same,} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{5.11}$$

Finally, it is of interest to compare the ranks of ρ_{alg} for AS and MMAS to both those of GBAS, and those of the expectations. These results were calculated to get a measure of the independence of each run. For instance, $\rho_{alg} = 0.7$ may begin as the worst setting for this parameter but then as m gets larger it becomes better than the other settings for ρ_{alg} (see Figure 5.12).

The expected outcome from GBAS is that as ρ_{alg} tends to 1, the performance will improve, as well as there being an improvement as m tends to ∞. This expectation is based on the following assumption:

Assumption: the problems are not deceptive, therefore an increase in knowledge about previous solutions is helpful with respect to the current search window.

When $m = n$, from Table 5.10 in most cases this is indeed the result, and Figure 5.13(a) illustrates an example of this for the TSP.

For problems eil76 and kroA100, in Table 5.10 the ranking is different from expectations with 0.99 being the worst ρ_{alg} setting. The reasons for this behaviour could be the size of the problems, or a landscape feature. However, if it was the size of the problem

one would expect the larger QAP and TS problems to show this same ranking, which they do not. Therefore, it is probably due to a feature of the problem landscape.

Both AS and MMAS have fairly mixed results, and no one combination is dominant. It can be seen that the value of 0.99 for ρ_{alg} is not favourable in either case ranking at either 3 or 4 (a result confirming Section 5.4).

In Table 5.11 the ranks of ρ_{alg} for the variable I_{GB} are shown. For GBAS, it is clear that as $\rho_{alg} \to 1$ the convergence of the algorithm is delayed as expected in the order $(0.7 < 0.9 < 0.95 < 0.99)$. The results for AS and MMAS are less clear and provide the researcher with an unpredictable parameter landscape to tune.

It is clear that m is a variable that can take a host of values. From the graphs on the left-hand side of Figure 5.13 one can see that each ρ_{alg} value draws a linear trend indicating an exponential decay (due to the log x-axis). This implies that after a certain point having larger values of m will have little performance value.

Table 5.12 shows the stability of the rankings at $m = n$ over a range of m values. These figures show how independent the results for each ρ_{alg} are. Immediately clear is the stability found in the column of GBAS. Most of the results for both performance and convergence are above 80%, with the exception of the larger TSP problems which are more stable for AS and MMAS. The hardest algorithm to predict appears to be MMAS which varies a great deal from its rankings at $m = n$. This is illustrated in Figure 5.13(e) and 5.13(f) which show how chaotic the variances are for low values of m, and that even as m tends to n the ρ_{alg} values below 0.99 are weaved together.

In Table 5.15 the percentage of points resulting from AS and MMAS that are in the same rank as those produced from the GBAS runs are given. This table shows the similarity between the algorithms. There are two points to notice from this table, the first is that the results for I_{GB} are higher than those of P_{GB}. Secondly, on average the results for Ant System are greater than those of MMAS for both variables. The conclusion that can be drawn from these trends is that in general the performance of ρ_{alg} is less predictable between GBAS and the other algorithms than convergence is. Although for both variables Ant System has more in common with GBAS than MMAS does.

In Table 5.13 the results for the comparison of performance between the expected ρ_{alg} rankings and each algorithm are given. It is evident that GBAS is much more likely than either of the other algorithms to conform to this expected ordering with 81%

of total points agreeing. Of this percentage it is the TSP that has the lowest value, approximately 54% of points.

The analogous table for I_{GB} is given in Table 5.14. Here GBAS agrees 100% with the expected ordering, suggesting that $\rho_{alg} = 0.99$ is the best setting for delaying convergence. For AS and MMAS, in that order, the results are lower, although AS has some problems that comply with the expected ordering completely. Therefore one would expect AS to correlate well with GBAS, and MMAS less so.

The correlations of both the algorithms are high. Using the Pearson Product-Moment Correlation test on {GBAS, AS} and {GBAS,MMAS} it was found that for P_{GB} the correlations were significant (see Table 5.16). For the variable I_{GB} the situation was not so distinct and are shown in Table 5.17. Furthermore, from Tables B.13 and B.14 the only generalisation that can be made is that Ant System is best correlated with GBAS when $\rho_{alg} < 0.95$. The number of results significantly correlated to GBAS for Ant System is 43%; for MMAS there is little correlation, with only 25% of results being significantly correlated to GBAS.

From these correlations the hypothesis tests result in the following statements:

- For P_{GB}, GBAS and AS are significantly correlated, reject Null Hypothesis.
- For P_{GB}, GBAS and MMAS are significantly correlated, reject Null Hypothesis.
- For I_{GB}, GBAS and AS are not significantly correlated, reject Alternative Hypothesis. Most likely to be correlated for $\rho_{alg} \in \{0.7, 0.9\}$.
- For I_{GB}, GBAS and MMAS are not significantly correlated, reject Alternative Hypothesis.

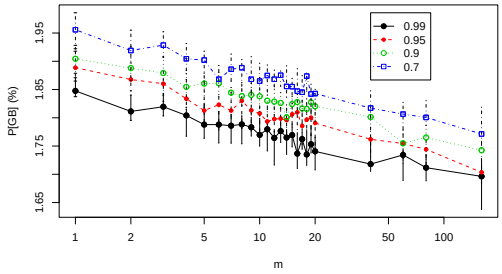
From these statistical test conclusions it can be seen that the results for P_{GB} of both AS and MMAS behave very similarly to GBAS when m is varied. However, there seems to be less guarantee in terms of their convergence. There are indications that the ρ_{alg} groups set up in Section 5.4 may also play a part in how well GBAS models AS. In the next section the variation of these parameters will be investigated in a slightly different way to try and get a better understanding of their relationship.

Problem	GBAS				AS				MMAS			
	0.7	0.9	0.95	0.99	0.7	0.9	0.95	0.99	0.7	0.9	0.95	0.99
gr17	4	3	2	1	3	2	1	4	3	1	1	4
bayg29	4	3	2	1	3	2	1	4	1	2	2	4
brazil58	4	3	1	2	3	2	1	4	1	2	3	4
eil76	3	2	1	4	3	2	1	4	1	2	3	4
kroA100	3	2	1	4	3	2	1	4	1	2	3	4
had18	4	3	2	1	4	2	1	3	3	2	1	4
sko81	4	3	2	1	3	2	1	4	2	1	3	4
tai50a	4	3	2	1	4	2	1	3	2	1	3	4
lipa80a	4	3	2	1	4	2	1	3	3	2	1	4
esc16c	4	3	2	1	4	2	1	2	1	2	2	4
bur26f	4	3	2	1	4	2	1	3	3	2	1	4
tai15b	4	3	2	1	4	3	1	2	1	2	3	4
tai100b	4	3	2	1	2	1	4	3	2	3	1	4
talent_10_10_5	4	3	2	1	4	3	1	2	1	1	1	4
talent_10_15_8	4	3	2	1	4	2	1	3	2	4	3	1
talent_10_20_5	4	3	2	1	3	2	1	4	2	1	3	4
talent_10_30_4	4	3	2	1	3	2	1	4	1	2	3	4
talent_10_100_4	4	3	1	2	3	2	1	4	3	2	1	4

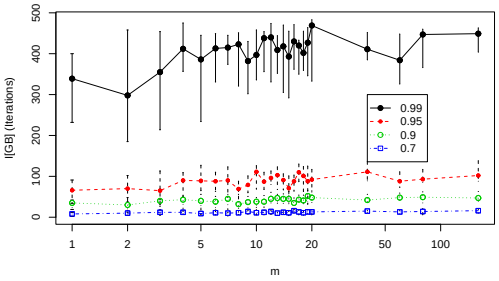
Table 5.10: Table showing the ranks of ρ_{alg} for P_{GB} when $m = n$. (1=closest to optimal)

Problem	GBAS				AS				MMAS			
	0.7	0.9	0.95	0.99	0.7	0.9	0.95	0.99	0.7	0.9	0.95	0.99
gr17	1	2	3	4	1	2	3	4	2	1	3	4
bayg29	1	2	3	4	1	2	4	3	2	1	3	4
brazil58	1	2	3	4	1	2	4	3	2	3	3	1
eil76	1	2	3	4	1	2	4	3	4	1	3	2
kroA100	1	2	3	4	1	3	4	2	4	2	1	3
had18	1	2	3	4	1	2	4	3	1	3	2	4
sko81	1	2	3	4	1	2	4	3	2	3	4	1
tai50a	1	2	3	4	1	2	4	3	2	4	3	1
lipa80a	1	2	3	4	1	2	4	3	2	4	3	1
esc16c	1	2	3	4	1	2	4	3	2	1	3	4
bur26f	1	2	3	4	1	2	4	3	3	1	2	4
tai15b	1	2	3	4	1	2	3	4	1	2	3	4
tai100b	1	2	3	4	1	4	3	2	1	3	4	2
talent_10_10_5	1	2	3	4	1	2	3	4	3	2	1	4
talent_10_15_8	1	2	3	4	1	2	3	4	2	3	1	4
talent_10_20_5	1	2	3	4	1	2	4	3	1	2	3	4
talent_10_30_4	1	2	3	4	1	2	4	3	3	2	1	4
talent_10_100_4	1	2	3	4	1	2	4	3	1	4	2	3

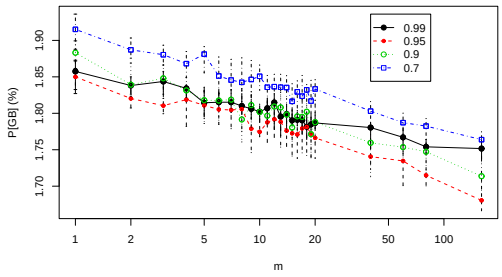
Table 5.11: Table showing the ranks of ρ_{alg} for I_{GB} when $m = n$. (4=latest convergence)



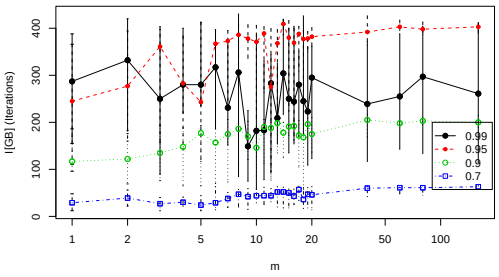
(a) P_{GB} using GBAS



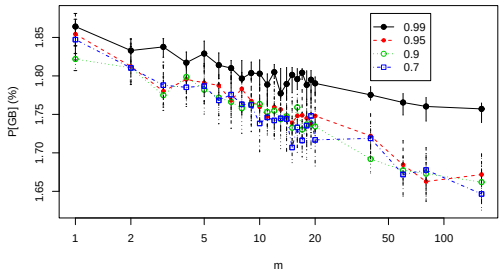
(b) I_{GB} using GBAS



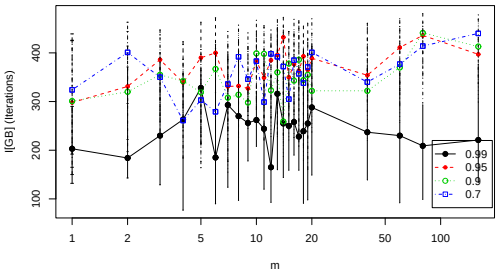
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 5.13: Example of the results achieved by varying ρ_{alg} with m over GBAS, AS and MMAS, in this case for the QAP problem lipa80a. (Log x-axis is used for m.)

Problem	P_{GB} Stability			I_{GB} Stability		
	GBAS	AS	MMAS	GBAS	AS	MMAS
gr17	100	89	6	100	50	22
bayg29	100	97	7	100	90	7
brazil58	39	100	56	100	56	6
eil76	9	100	83	100	87	9
kroA100	14	93	100	100	69	3
had18	95	11	11	100	53	32
sko81	92	25	42	100	88	33
tai50a	94	51	22	100	90	24
lipa80a	100	42	8	100	75	21
esc16c	88	6	18	100	6	24
bur26f	93	81	19	100	78	11
tai15b	88	56	19	100	38	13
tai100b	72	38	14	100	28	7
talent_10_10_5	91	45	9	100	100	9
talent_10_15_8	94	75	13	100	81	13
talent_10_20_5	100	14	10	100	48	29
talent_10_30_4	100	48	16	100	68	13
talent_10_100_4	10	52	38	100	97	3

Table 5.12: Table showing the stability of the rankings of ρ_{alg} when using GBAS, AS and MMAS. (Percentages are given as integers.)

Algorithm	Min.	Q1	Median	Mean	S.d.	Q3	Max.
GBAS	3.45	87.68	92.13	81.36	29.90	98.68	100.00
(TSP)	3.45	4.35	61.11	53.78	48.23	100.00	100.00
(QAP)	72.41	88.05	92.13	90.16	8.17	94.27	100.00
(TS)	89.66	90.91	93.75	94.86	4.92	100.00	100.00
AS	0.00	0.00	0.00	5.51	12.21	0.00	43.75
(TSP)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
(QAP)	0.00	0.00	0.00	8.99	15.54	12.31	43.75
(TS)	0.00	0.00	0.00	5.46	12.20	0.00	27.27
MMAS	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 5.13: Table showing how, for each algorithm, the rankings of ρ_{alg} compare to the expected rankings, which is $(0.7, 0.9, 0.95, 0.99)$, for P_{GB}. The statistics have been broken down by problem type where necessary.

Algorithm	Min.	Q1	Median	Mean	S.d.	Q3	Max.
GBAS	100.00	100.00	100.00	100.00	100.00	100.00	100.00
AS	0.00	0.00	10.06	28.62	34.94	49.34	100.00
(TSP)	0.00	0.00	0.00	10.67	22.04	3.33	50.00
(QAP)	0.00	0.00	10.06	24.89	33.47	39.97	94.12
(TS)	0.00	29.03	52.38	52.53	39.97	81.25	100.00
MMAS	0.00	1.09	10.10	12.50	11.75	19.96	33.33
(TSP)	0.00	4.35	11.11	13.09	11.34	16.67	33.33
(QAP)	0.00	0.00	3.70	8.80	11.34	14.64	29.41
(TS)	6.45	9.09	13.79	17.83	11.38	28.57	31.25

Table 5.14: Table showing how, for each algorithm, the rankings of ρ_{alg} compare to the expected rankings, which is $(0.7, 0.9, 0.95, 0.99)$, for I_{GB}. The statistics have been broken down by problem type where necessary.

Problem	PRO(AS) P_{GB}	PRO(MMAS) P_{GB}	PRO(AS) I_{GB}	PRO(MMAS) I_{GB}
gr17	3	21	75	51
bayg29	1	8	50	55
brazil58	10	1	39	32
eil76	37	10	47	33
kroA100	47	23	33	17
had18	49	8	74	54
sko81	25	14	47	16
tai50a	38	13	48	23
lipa80a	40	13	56	30
esc16c	60	12	97	59
bur26f	30	20	44	39
tai15b	69	25	69	36
tai100b	15	11	39	18
talent_10_10_5	61	34	100	30
talent_10_15_8	33	22	91	48
talent_10_20_5	23	21	76	58
talent_10_30_4	14	12	65	48
talent_10_100_4	14	13	50	37
Mean	32	16	61	38

Table 5.15: Table showing the similarity of the ranks of the points between AS/MMAS and GBAS. (Cor=Pearson Product-Moment Correlation, PRO(X)=Percentage of points from the running of algorithm X that are in the same rank as in the corresponding GBAS runs given as integers.)

Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
TSP	20	20	20
QAP	32	32	32
TS	20	20	20

Table 5.16: Table summarising the hypothesis test results for P_{GB} varying m. (For more detailed results refer to Tables B.11 and B.12.)

Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
TSP	20	11	7
QAP	32	11	8
TS	20	9	3

Table 5.17: Table summarising the hypothesis test results for I_{GB} varying m. (For more detailed results refer to Tables B.13 and B.14.)

5.7 Variation in m_2-Values

When considering the question of how many ants is a good number to assign to the parameter m, besides the direct experiment (as seen in the previous experiment), it is also necessary to consider how much useful work is being done by those ants. This is known as the *efficacy* of the sampling. This is very important as the construction method used to build the solutions takes longer than other algorithms, such as Tabu Search or Simulated Annealing, and needs to be used for the benefit of the Ant algorithm.

Therefore in this section the number of samples is limited to 1000 and the number of iterations is determined by Equation 5.12.

$$1000 = max_{it} \cdot m \tag{5.12}$$

The number of 1000 was chosen for two reasons. The first was the practical consideration of speed as there were many experiments to perform. The second, which was the more important, was that it enabled at least 10 iterations to be performed. This was considered a reasonable amount of time for the effects of the update rule to affect the matrix.

As for the previous experiment the following conditions held for all experiments.

- $\alpha = 1$
- $\beta = 0$
- $\rho_{alg} \in \{0.7, 0.9, 0.95, 0.99\}$

For each test m was varied as follows:

- for $n <= 50$, $m \in \{1, \ldots, n\}$
- for $n > 50$, $m \in \{1, \ldots, 0.25n, 0.5n, n, 2n\}$.

In Figure 5.14 an example of the results for the TSP problems is given. There is nothing unexpected in the I_{GB} graphs with their curve decaying as m tends to $2n$. However, for P_{GB} there are a number of observations to be made.

First, both GBAS and AS show the possibility for optimal m values that maximise the efficacy for a problem. For GBAS, as ρ_{alg} reduces, this value both increases but also

becomes more robust. This is not unexpected but confirms the view that a low ρ_{alg} should be compensated for by a higher number of ants. In the case of MMAS the best results are always achieved at very low number of ants. This confirms the view found in previous research that low m performs very well for MMAS.

In Figure 5.15 the same results are shown for the QAP data set. In general, the results for all the algorithms are bunched together so that the actual ρ_{alg} value is not significant. For GBAS, the results are more chaotic, but are similar to their TSP counterpart. However, AS is more steady, favouring values between 5 and 20 ants. For MMAS, the results are best for lower m values, although as m tends to 20 the difference between ρ_{alg} values becomes harder to distinguish.

Finally in Figure 5.16 the graphs for a Talent Scheduling problem are given and show similar results to those for the TSP. The wider distributions shown by the error bars are due to the y-axis being the global best solution objective values and not the percentage from the optimums.

From these sets of results the derivative graphs can be built to see how the lowest median m values vary with n. These are particularly enlightening and show a number of phenomena.

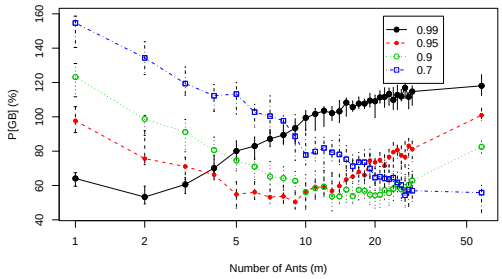
- For all domains and algorithms, as ρ_{alg} tends to 0 the number of ants required to maintain performance increase.
- QAP is the most volatile of the domains with no clear trends for AS and MMAS.
- For all domains, GBAS requires approximately the same number of ants for all values of n, and is more dependent on ρ_{alg}.
- For $\rho_{alg} > 0.9$, 1 to 10 ants is sufficient for all domains. It appears that pheromone updates are more important than sampling of the neighbourhood for the algorithms.

From these results it is possible to conclude that n is not the best value to be sampling with at high ρ_{alg} values. As $\rho_{alg} \to 0$ the problem may require as much as $2n$ ants for best performance.

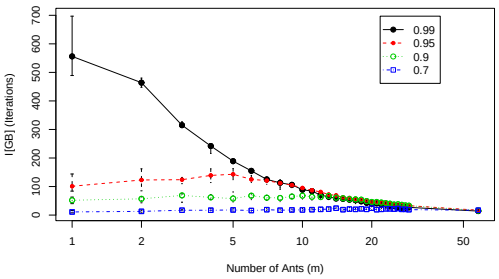
In general two conclusions can be reached with regard to the parameter m:

1. For all three algorithms, in a practical setting m can be regarded as independent of n.

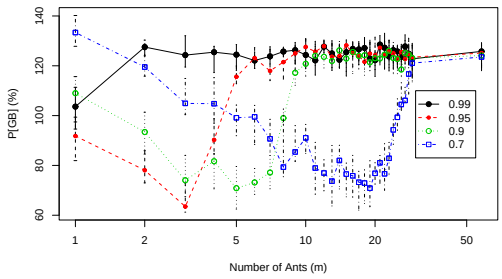
2. For all three algorithms, 1 to 10 ants will get approximately the same performance for $\rho_{alg} > 0.9$. For $\rho_{alg} = 0.9$ a value of 20 is sufficient and then for each reduction of ρ_{alg} by 0.1, m should be doubled.



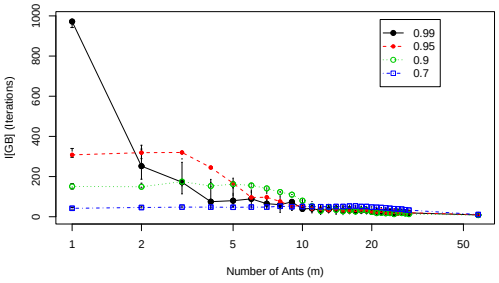
(a) P_{GB} using GBAS



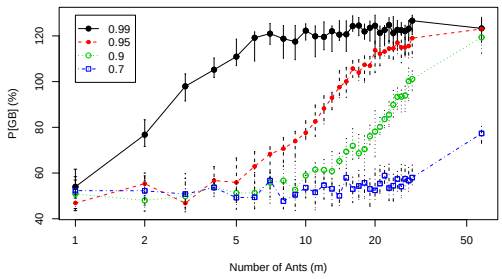
(b) I_{GB} using GBAS



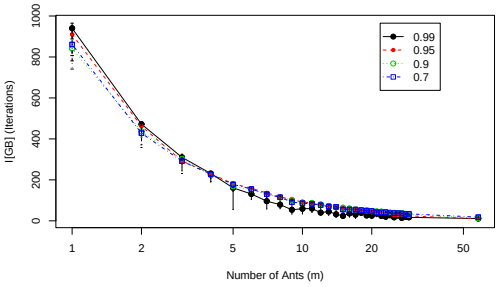
(c) P_{GB} using AS



(d) I_{GB} using AS

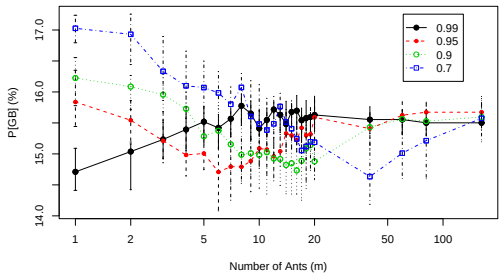


(e) P_{GB} using MMAS

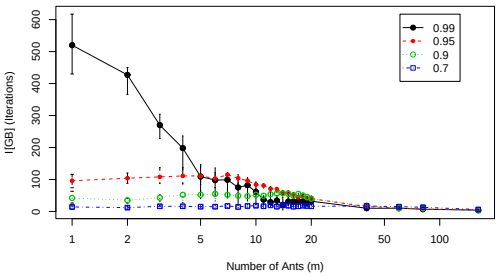


(f) I_{GB} using MMAS

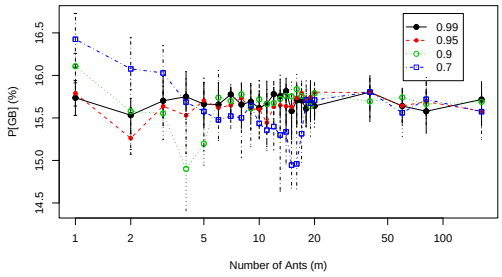
Figure 5.14: Example of the results achieved varying m, but limiting the runs to a maximum of 1000 samples, over GBAS, AS and MMAS on the TSP data set, in this case for bayg29. (Log x-axis is used for m.)



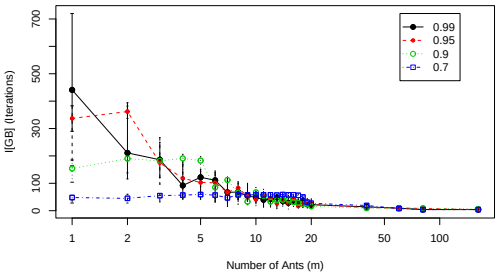
(a) P_{GB} using GBAS



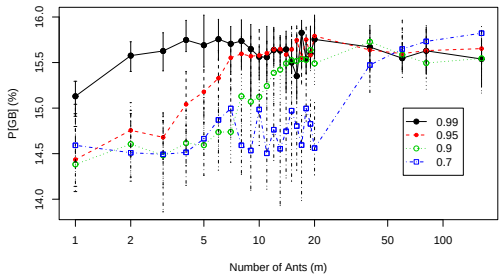
(b) I_{GB} using GBAS



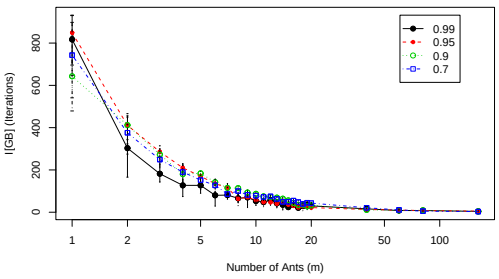
(c) P_{GB} using AS



(d) I_{GB} using AS

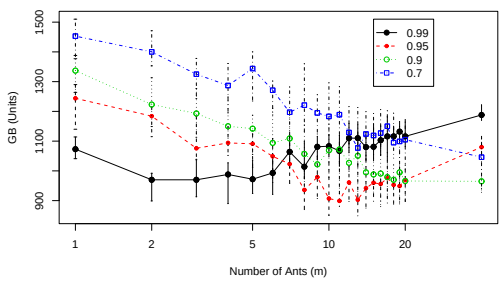


(e) P_{GB} using MMAS

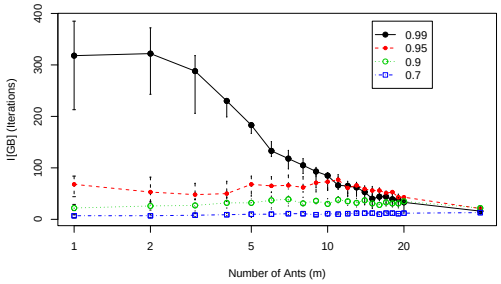


(f) I_{GB} using MMAS

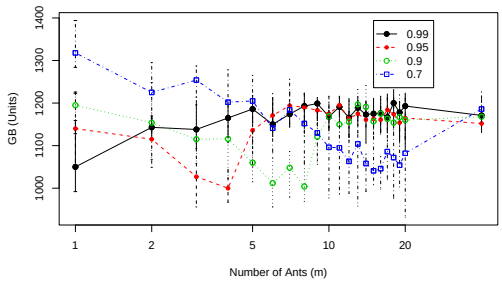
Figure 5.15: Example of the results achieved varying m, but limiting the runs to a maximum of 1000 samples, over GBAS, AS and MMAS on the QAP data set, in this case for sko81. (Log x-axis is used for m.)



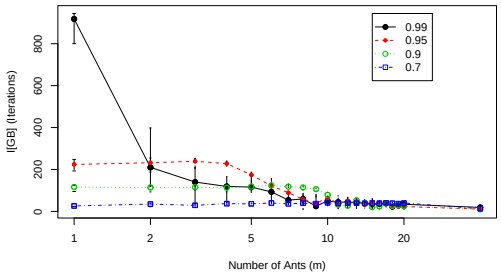
(a) P_{GB} using GBAS



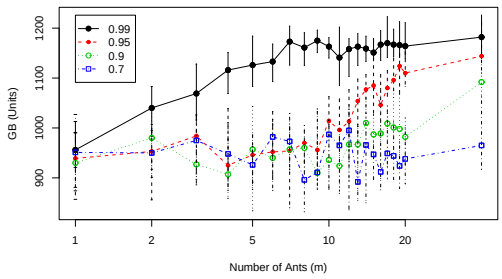
(b) I_{GB} using GBAS



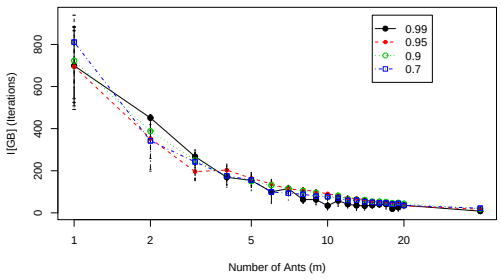
(c) P_{GB} using AS



(d) I_{GB} using AS

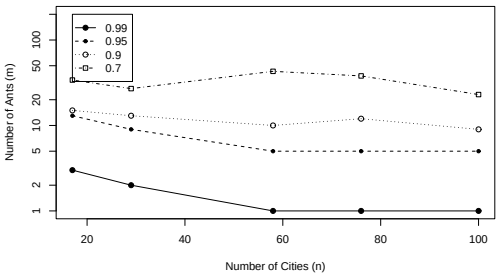


(e) P_{GB} using MMAS

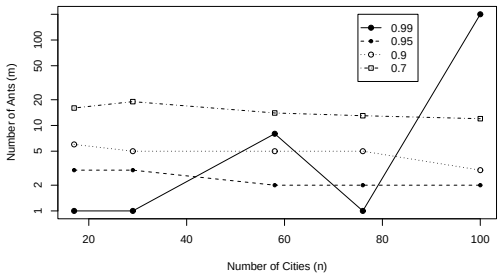


(f) I_{GB} using MMAS

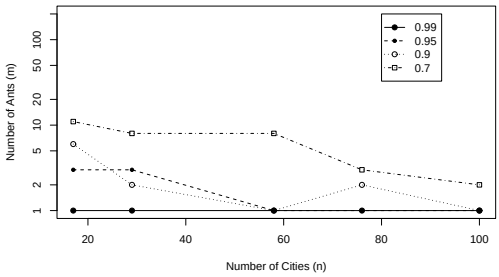
Figure 5.16: Example of the results achieved varying m, but limiting the runs to a maximum of 1000 samples, over GBAS, AS and MMAS on the TS data set, in this case for talent_10_20_5. (Log x-axis is used for m.)



(a) TSP using GBAS

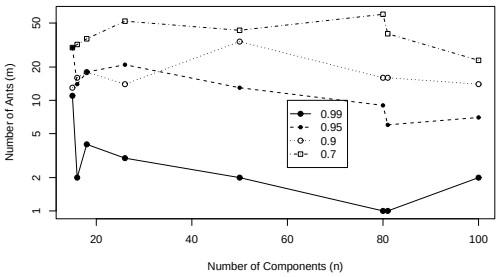


(b) TSP using AS

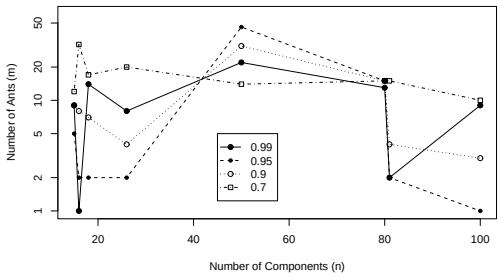


(c) TSP using MMAS

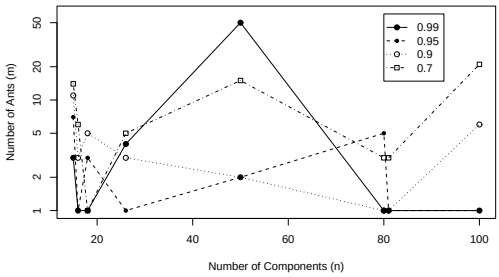
Figure 5.17: Figures for the TSP data set showing how the optimum m varies with the number of cities, plotted on log(y)-x axes.



(a) QAP using GBAS

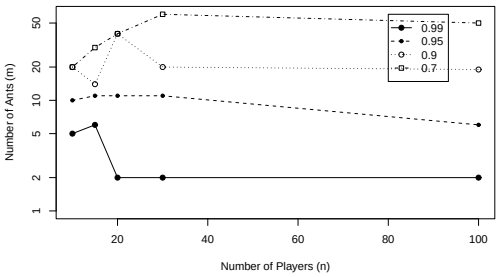


(b) QAP using AS

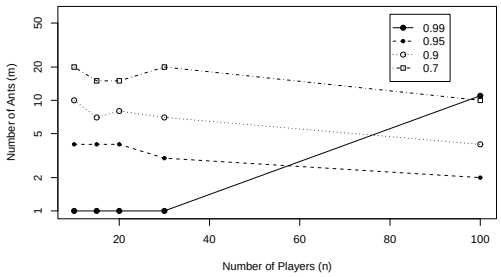


(c) QAP using MMAS

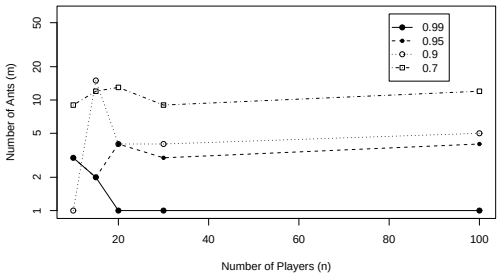
Figure 5.18: Figures for the QAP data set showing how the optimum m varies with the number of components in the problem, plotted on log(y)-x axes.



(a) TS using GBAS



(b) TS using AS



(c) TS using MMAS

Figure 5.19: Figures for the TS data set showing how the optimum m varies with the number of pieces, plotted on log(y)-x axes.

5.8 Conclusions

The goal of this chapter was to find out if the Graph-based Ant System algorithm (GBAS) was a good model of how Ant System (AS) and the Max-Min Ant System (MMAS) worked by using empirical testing to find correlations between the three algorithms. Each algorithm was stripped of its enhancements leaving only the basic structure that still enabled it to solve problems. The algorithms were then compared under the following assumptions:

1. The parameters n_0 and q_0 can be removed and compensated for by less aggressive settings of the other algorithm parameters.
2. The parameters involved in the probability rule are independent of those in the trail update rule.
3. The heuristics combined with the algorithms returned values that indicated their true impact on the solution being generated.
4. The problems used are not deceptive, therefore the knowledge gained from one solution can be used to help generate better solutions.

Under these assumptions various experiments were performed and the following conclusions were reached.

- There is no difference in using a direct multiple ρ or a weighted sum approach in the update rule for Ant System.
- The performance characteristics when varying ρ_{alg} can be split into two, one for the range $(0, 0.9]$ and one for $[0.9, 1)$. In the lower range, GBAS and AS are significantly correlated; GBAS and MMAS are significantly correlated in terms of performance but not in terms of convergence. In the higher range, neither AS or MMAS are significantly correlated to GBAS.
- The optimal values for α and β, given these problems, is 1 and 2 respectively.
- Whether α and β for various algorithms are correlated depends on the problem: for TSP they are correlated, for TS the situation is mixed, and for QAP they are not. In general, increased α produces more insignificant correlations, and increased β produces more significant correlations.
- When varying the number of ants the algorithms are correlated with GBAS only

in terms of performance, and that for convergence they are uncorrelated.

- The best number of ants to run with each algorithm is independent of the size of the problem but dependent on ρ_{alg}, values between 1 and 10 are acceptable for high ρ_{alg} but as it tends to 1, this number should be doubled with each decrement of 0.1.

These conclusions show that, in general, GBAS is a good model for talking about the other two algorithms, given the parameters are not set to extreme values. They show how different variables depend on different aspects, for instance some are problem dependent while others are algorithm dependent. In terms of ρ_{alg}, the picture has been shown to be more complicated than expected, and that as it tends to 1 this is where the different matrix representations show their flaws and benefits.

The experiments have shown clearly that making the number of ants per iteration equal to the size of the problem is unwarranted for high ρ_{alg} and that better performance can be achieved with fewer ants. The critical focus should be the number of pheromone updates, not the number of ants per run. However, the assumptions made upon the problems chosen might interfere with this prediction.

It is clear that future work should concentrate on removing the assumptions about the domains and heuristics. It would also be beneficial to use harder sets of problems than the TSPLIB and QAPLIB benchmarks where one can relate the problem structure with the algorithm. This would allow studies to identify whether various matrix structures are better at learning different landscape characteristics.

5.9 Summary

This completes the chapter on the suitability of GBAS for talking about other Ant algorithms. In the next chapter the assumption that the heuristics are generally valid, will be challenged and the reaction of the algorithms will be analysed. Following this the algorithms will be combined with a number of local search methods.

Chapter 6

Heuristics as a Source of Knowledge in Ant Algorithms

In the previous chapter the Graph-based Ant System was compared to Ant System and the Max-Min Ant System. In these experiments the heuristics used were assumed to be an accurate guide to the search space. In this chapter this assumption is removed and the following question is asked, "how well do Ant algorithms handle misleading information given to them by a heuristic?"

This chapter starts with a brief introduction as to why this question is important, after which the various varieties of heuristic will be discussed. Following this, in Section 6.2, there will be some general detail about how the experiments were carried out and then the results will be discussed. At the end of the chapter are the final conclusions.

6.1 Introduction

It may seem strange to investigate what happens when an algorithm is fed misinformation. The natural question is to ask why anyone would feed the algorithm misleading information in the first place. The answer is that no heuristic is 100% accurate at guiding the algorithm to the correct area of the search space, if it were the problems would become trivial. Given this inaccuracy, it is important to know what happens when the heuristic is wrong. The response of a good algorithm that is intelligent and adaptable, as is claimed for Ant algorithms, is that the algorithm should identify the source of misleading information, and then take action to remove this source from use.

It is also the case that when adapting an algorithm to a new application it is not always obvious what a good heuristic should look like, and a researcher may try many different *hunches* to come up with a suitable heuristic. However, even given this work done beforehand, there is still no way to guarantee that for every new problem in that domain the new heuristic will work, therefore it is important for the algorithm to be able to adapt.

In this chapter the situation where misleading information is given is taken to extremes to find out how resilient the three Ant algorithms are. GBAS, AS and MMAS are all being tested because it is unclear how each pheromone matrix structure will react to the introduction of misinformation. It is expected that the results will not be encouraging, as in Section 5.5 it was shown that even when low β values are used, the algorithm comes to rely on the heuristic. The five heuristics that will return the misinformation are as follows:

- Reverse Cost Matrix (RCM),
- Static Random Cost Matrix (SRCM),
- Dynamic Random Cost Matrix (DRCM),
- Fixed Random Increment (FRI),
- Dynamic Random Increment (DRI).

Figure 6.1 shows all the misleading heuristics in a diagram that places them between the best heuristic used and what is expected to be the worst heuristic. In this figure the heuristics are divided into *static* and *dynamic* heuristics to simulate those that only depend on problem dependent information, and those that may depend on previous iterations respectively.

Starting with the worst heuristic at the bottom of the figure, which is the *Reverse Cost Matrix* heuristic (RCM). This heuristic is defined as the reverse of the cost matrix of the problem. Therefore, the largest cost is assigned to the arc with the smallest cost and vice versa. This means that the algorithm should be guided constantly to very bad solutions.

Above this are the two most random value heuristics. The first to the left is the *Static Random Cost Matrix* heuristic (SRCM). This heuristic is defined as a fixed matrix, the same size as the normal cost matrix, filled with random values between zero and the

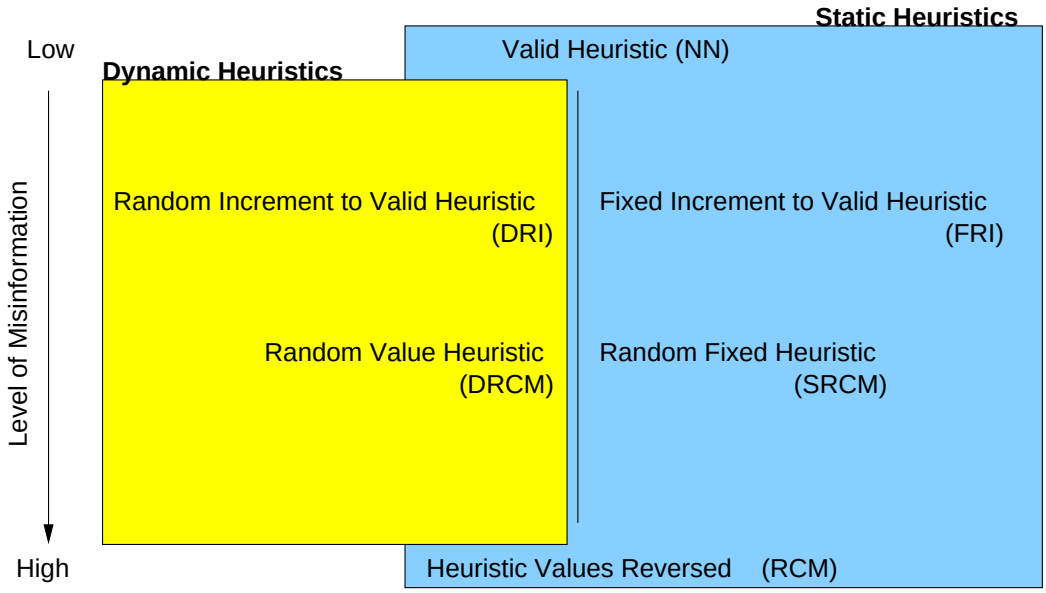


Figure 6.1: Figure showing the hierarchy of the various heuristics used to feed the Ant algorithms with misleading information.

largest cost in the problem (c_{max}, where the matrix C is the cost matrix of the problem). In the dynamic version (DRCM) the returned value is simply a random integer in the same range, therefore there is always the possibility of the correct cost being chosen given enough time.

On the next level, a random increment in the range $[0, c_{max}]$ is chosen to be added or subtracted with a probability of 0.5 to the existing cost of the particular arc ($c_{ij} +/- rand(1, c_{max})$). This heuristic is called *Fixed Random Increment* (FRI). In the static version the same increment is used throughout a single run for all arcs. In contrast, the dynamic version (DRI) generates a new increment every time the heuristic is called. The consequence of these increments is that the relative values of the cost matrix should be kept reasonably intact. If these values were only added, or subtracted, exclusively there would be no change to the performance of the algorithm. The key is that there is some small variation in the distribution of values returned by the heuristic.

Finally the best heuristic of the group, Nearest Neighbour (NN), is used as a comparison of performance. For the Talent Scheduling problems the Manhattan Distance heuristic, introduced in Section 3.3.3, will be used, but referred to as the Nearest Neighbour heuristic.

The range of values was set at between zero and the maximum value in the cost matrix C, because this was the same range offered by the Nearest Neighbour heuristic, and

any other might have added unwanted bias.

6.2 Experimental Methodology

To carry out these experiments, conditions were kept similar to those in the previous chapter. The parameters that were chosen to be manipulated were:

- $\alpha \in [0,5]$
- $\beta \in [0,5]$
- $\rho_{alg} \in \{0.5, 0.7, 0.9, 0.99\}$

α and β were chosen as these parameters clearly alter the influence the heuristic in the algorithm, and were varied in the range $[0,5]$. Unlike in the previous chapter the heuristic and ρ are not assumed to be independent, therefore it was decided to vary ρ_{alg} as well. This assumption was removed because it is expected that low ρ_{alg} values will do well, as they should allow the algorithm to get rid of misleading information quicker, as opposed to higher values of ρ_{alg}, which would decay the pheromone intensities more slowly. Therefore high values of ρ_{alg} are expected to converge earlier with worse quality solutions. m is not varied and is kept at a value of 10, a value chosen as a consequence of the results in the previous chapter. The other conditions were as follows:

- $max_{it} = 500$, a run was stopped after 500 iterations.
- $max_{ti} = 300$, 5 minutes maximum per run.
- Only global best solution are used to update the pheromone matrix.

Each run was repeated 25 times and for each of the 6 problems, creating a total of 21,600 runs (36 combinations * 4 algorithms * 25 trials * 6 problems). The number of problems was reduced from the 18 used in the previous chapter as there were so many runs to do, and as the heuristic was providing misleading information it was not expected the runs would find the optimum very often. Therefore the runs were expected to continue for the full 500 iterations, thereby increasing the total runtime to an unacceptable level. The 6 problems consisted of 3 TSP and 3 Talent Scheduling (TS) of varying sizes, randomly chosen from the respective sets. All three algorithms (AS,GBAS,MMAS) were run on all the problems.

- 3 TSP Problems: ulysses16, brazil58, rd100
- 3 TS Problems: talent_10_20_6, talent_10_30_12, talent_10_100_2

As for the previous chapter the two variables being measured are P_{GB}, which indicates the solution with the best objective value found during the run (commonly referred to as the performance), and I_{GB}, which is the index of the last iteration that produced a better solution (commonly referred to as the convergence). Convergence is defined as being equal to I_{GB} for the reasons given in Chapter 5. In the graphs that will accompany the discussion of the results in this chapter there are no error bars for presentation reasons. The variance between the various trials was very similar and as it did not form the basis of any discussion, for reasons of clarity these were removed.

In this chapter the graphs have on their x-axis, integers from 1 to 36. These correspond to the pairs $(0,0)$ to $(5,5)$, where the coordinates are (α,β). As the scale increases β is the first to be incremented, thereby all the α values are grouped together. This very often gives the pattern of a peak at $\beta = 0$ and then a steady decline as $\beta \rightarrow 5$. To elucidate, the points on the x-axis at which the α values change, are as follows: $x = (1,7,13,19,25,31)$, and at these points $\beta = 0$.

6.3 Results

In this chapter the results are split into a number of subsections. This will aid reading, and will split them into coherent groups of experiments. Subsection 6.3.1 gives a general description of the results for each individual heuristic, relating what the performance attributes were expected to be and what actually occurred. Subsection 6.3.2 then compares each heuristic to the Nearest Neighbour heuristic. This is clearly necessary to compare how the Ant algorithms performed on the misleading heuristics relative to the more accurate heuristic. Finally, in Subsection 6.3.3 the dynamic heuristics are compared to the static heuristics.

6.3.1 Individual Heuristic Performance

For each of the TSP and Talent Scheduling domains an example figure is given for each heuristic. Each figure will show how the performance (P_{GB}), and convergence (I_{GB}) varied for each heuristic over the four values of ρ_{alg}.

In Figure 6.2 an example of the TSP results is given for the Nearest Neighbour heuristic. This shows how the algorithms should react given a good heuristic, which returns values that are indicative of better solutions. The graphs show peaks where β is zero, and then these decay as β is increased and performance gets better up to a value of five. As α increases the solution quality degrades slightly for all the algorithms.

GBAS tends to convergence latest for $\rho_{alg} = 0.99$, and the others group at values less than a hundred iterations. As α is increased, the number of iterations taken to converge should decrease. With Ant System there tends to be more iterations before convergence than for GBAS. Furthermore, all ρ_{alg} values appear reasonably independent of α and β. In contrast, MMAS has very high values for I_{GB} while $\alpha = 1$, but these values reduce as α is increased. For $\rho_{alg} < 0.99$, this reduction is immediate for $\alpha = 2$, whereas for the value 0.99 the reduction is more gradual, peaking whenever $\beta = 0$.

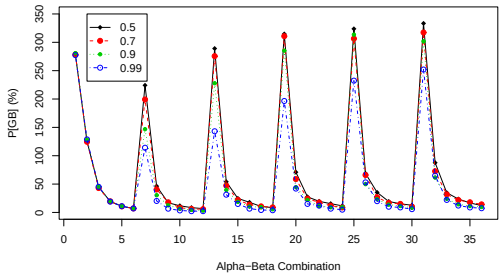
Figure 6.3 shows the analogous graphs for Talent Scheduling. For all three algorithms the highest peaks for those illustrating P_{GB} are when $\beta = 0$. In general the ranks of the various ρ_{alg} results are the same as for the TSP and are as expected ($(0.5 > 0.7 > 0.9 > 0.99)$ for GBAS, and $(0.99 > 0.5 > 0.7 > 0.9)$ for both AS and MMAS). The results for I_{GB} are very similar as in the previous figure for all the algorithms. The only exception is that there are fewer peaks at $\beta = 0$ to disturb the gradients of the lines, especially when $\rho_{alg} = 0.99$.

These two sets of figures give a benchmark for analysing the graphs from the other heuristics for the two problem domains.

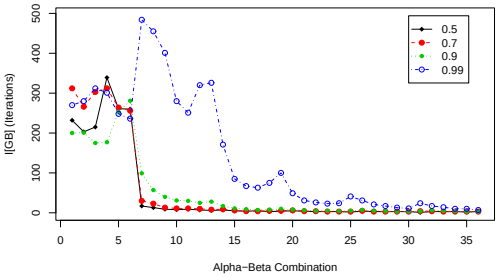
6.3.1.1 Dynamic Random Cost Matrix

The Dynamic Random Cost Matrix heuristic (DRCM) gives the Ant algorithm a random integer each time it requests a heuristic value. This, as with all the heuristics chosen, is an extreme case, simulating a poor knowledge of the problem landscape.

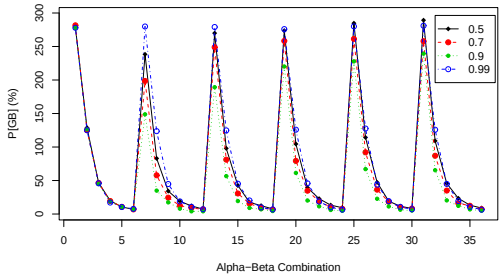
In Figure 6.4 the graphs show how the Ant algorithms performed on one of the TSP problems. The start of each P_{GB} graph shows that for $\alpha = 0$ the algorithms can make no gains, thus producing very poor quality solutions about 500% from the optimum. In all three algorithms, $\rho_{alg} = 0.99$ performs poorly, but when $\rho_{alg} < 0.99$, for GBAS and AS, there is improvement as $\beta \to 5$ for all α values, as opposed to MMAS which reacts in the opposite way. The scale of these improvements is approximately 200-250%.



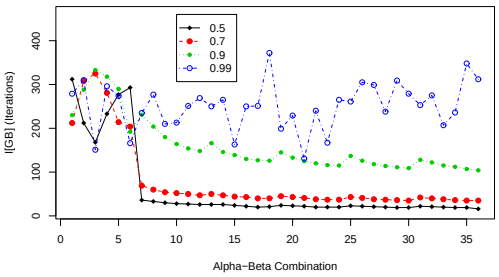
(a) P_{GB} using GBAS



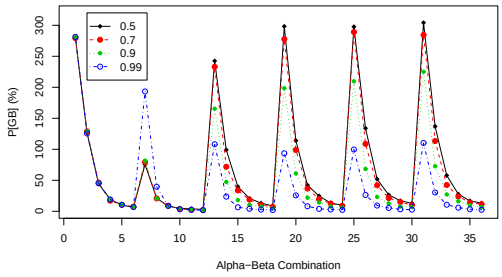
(b) I_{GB} using GBAS



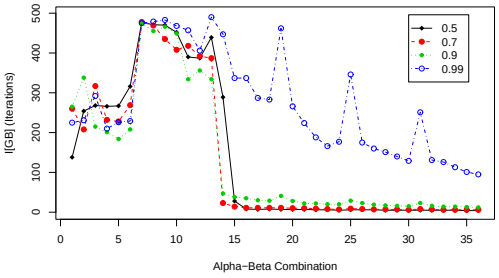
(c) P_{GB} using AS



(d) I_{GB} using AS

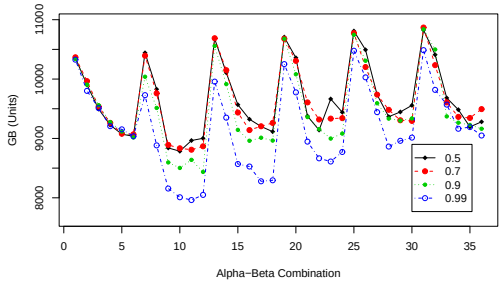


(e) P_{GB} using MMAS

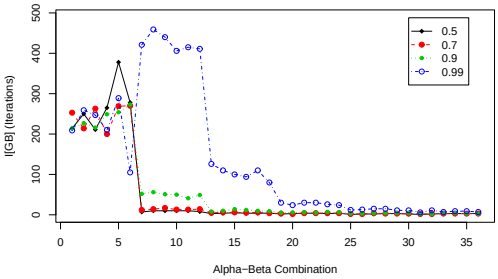


(f) I_{GB} using MMAS

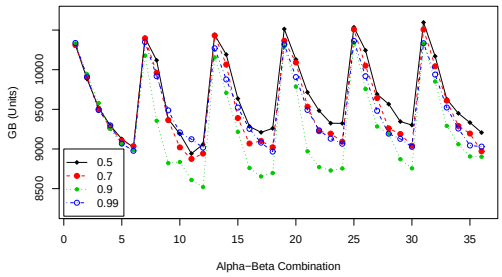
Figure 6.2: Example of the results achieved for the Nearest Neighbour (NN) heuristic for the TSP domain, problem Brazil58.



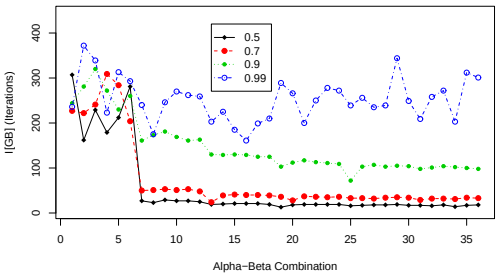
(a) P_{GB} using GBAS



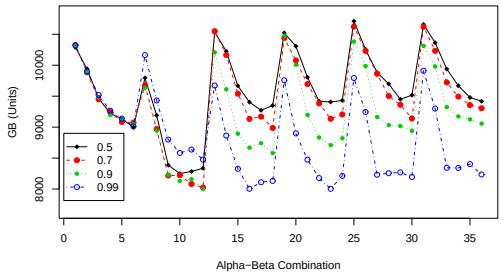
(b) I_{GB} using GBAS



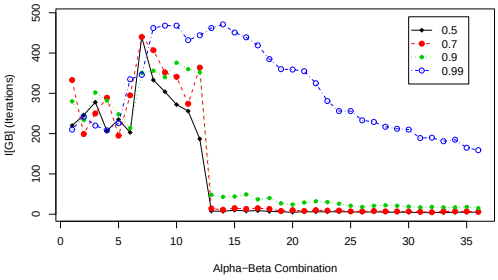
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 6.3: Example of the results achieved for the Nearest Neighbour (NN) heuristic for the Talent Scheduling domain, problem talent_10_100_2.

For MMAS, on the other hand, between $\beta = 0$ and $\beta = 5$ the solution quality gets worse by 250%. This shows that MMAS does not cope with the misinformation coming from this heuristic in the same way as GBAS and Ant System. In Figure 6.5(e) MMAS's performance shows troughs in the graph that occur when $\beta \geq \alpha$, and as β is increased, so the performance decreases. This shows there is some resilience to the heuristic as long as β and α are balanced.

The best convergence is achieved, for GBAS and Ant System, when ρ_{alg} is set high. For all three algorithms, the best quality solutions occur when the convergence is latest. Furthermore, the consistency of convergence between the various ρ_{alg} values, seen in Chapter 5, now has to be finely tuned.

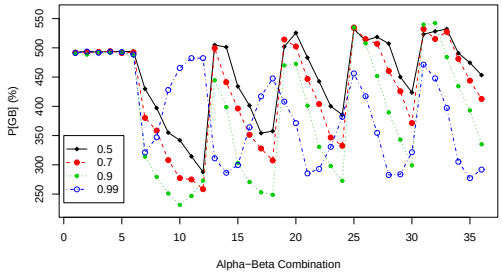
Figure 6.5 shows the results for one of the Talent Scheduling problems. In general, the graphs follow a similar pattern to those of the previous figure. For GBAS, in terms of performance, there is more distinction between the various ρ_{alg} values, most likely due to the difference in convergence times. $\rho_{alg} = 0.99$ finds the best quality solutions for GBAS, in contrast to Figure 6.4(a) for the TSP.

The performance of Ant System for the Talent Scheduling problem contrasts with the TSP in that when $\rho_{alg} = 0.99$, instead of being one of the worst performing values, it is now one of the better values, with the performance of $\rho_{alg} = 0.5$ sitting just above it. The algorithm's convergence shares similarities with the TSP in the trends of $\rho_{alg} \in \{0.5, 0.7, 0.99\}$, but is much more conservative when a value of 0.9 is used.

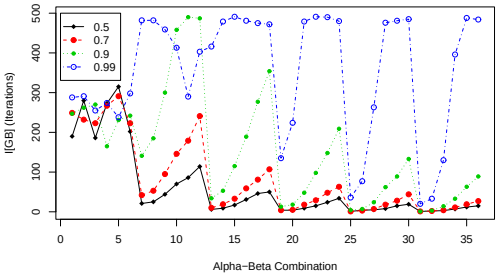
MMAS, for both P_{GB} and I_{GB}, shows similar trends to its TSP counterpart. For P_{GB}, as β increases so does the objective value of the best solutions found, the difference being that the quality is also dependent upon α. This is in contrast to the TSP graph, where all α values result in similar performance. The graphs in I_{GB} show the same rapid decrease as α increases, and then as $\beta \geq \alpha$ the number of iterations taken to converge increases rapidly from 0 to 300. This is indicative of the exploration that can occur when misleading heuristics are used. Furthermore, given the correct ρ_{alg} value this exploration does not have a detrimental effect on solution quality.

6.3.1.2 Reverse Cost Matrix

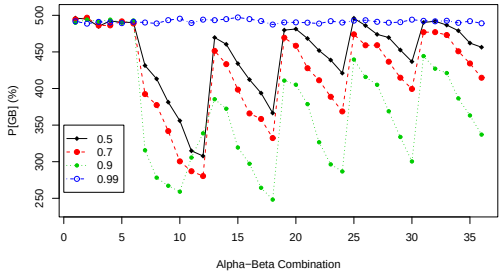
The Reverse Cost Matrix (RCM) heuristic is illustrated in Figure 6.6 for the TSP. This shows the strongest evidence for the shape of performance when a misleading heuristic



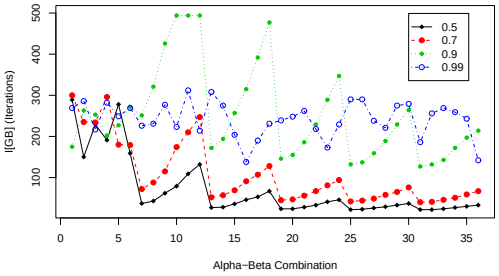
(a) P_{GB} using GBAS



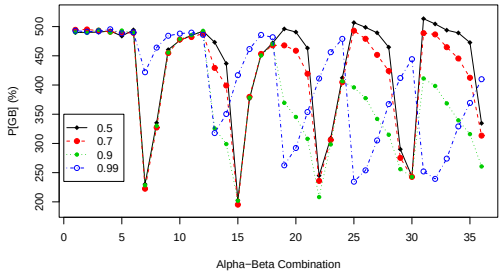
(b) I_{GB} using GBAS



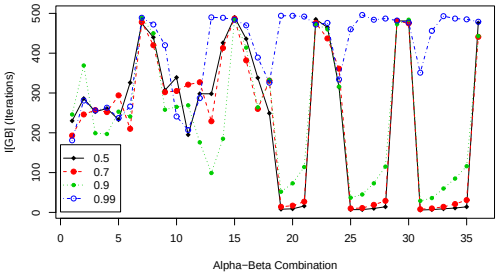
(c) P_{GB} using AS



(d) I_{GB} using AS

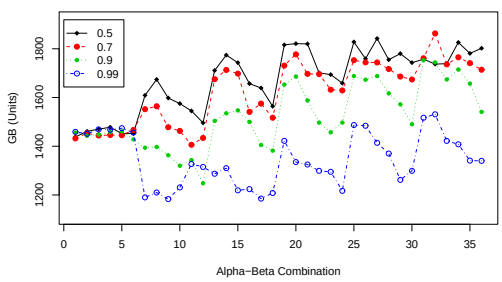


(e) P_{GB} using MMAS

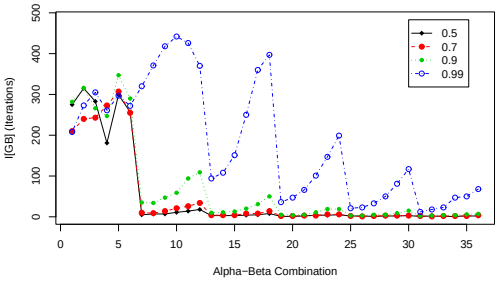


(f) I_{GB} using MMAS

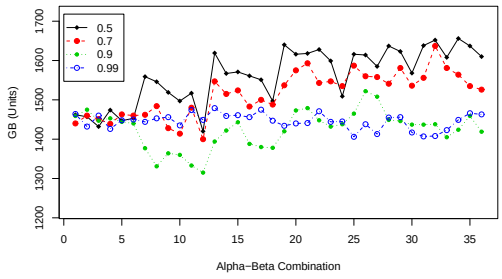
Figure 6.4: Example of the results achieved for the Dynamic Random Cost Matrix (DRCM) heuristic for the TSP domain, problem rd100.



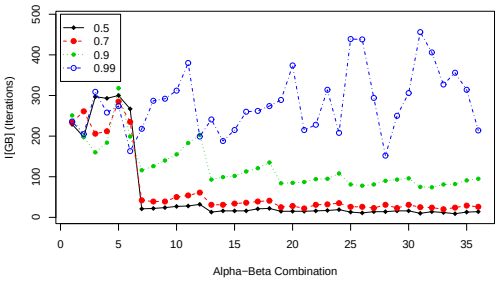
(a) P_{GB} using GBAS



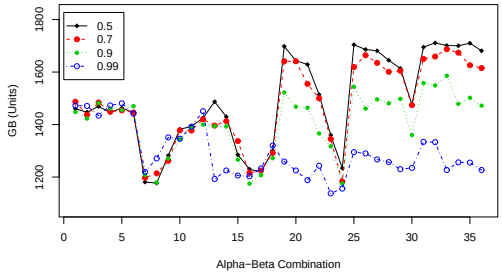
(b) I_{GB} using GBAS



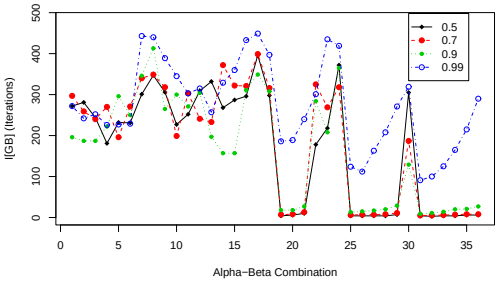
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 6.5: Example of the results achieved for the Dynamic Random Cost Matrix (DRCM) heuristic for the Talent Scheduling domain, problem talent_10_20_6.

is merged with an Ant algorithm. All three P_{GB} graphs show that as β increases, the performance degrades. Contrary to expectations, a value of 0.9 or 0.99 for ρ_{alg} still produces the best results. On the other hand, for MMAS when $\alpha = 1$ all values of ρ_{alg} are equally successful, but this diminishes as α increases, with the lower values of ρ_{alg} unable to compete.

The convergence increases for GBAS and Ant System as $\rho_{alg} \rightarrow 1$, while the other settings remain at very low values. The I_{GB} graph for Ant System shows the same convergence properties as one would expect for the Nearest Neighbour heuristic. In Figure 6.6(f), the convergence of MMAS is high when $\alpha < 4$, but after this, for $\rho_{alg} < 0.99$, there are very few iterations performed before convergence occurs.

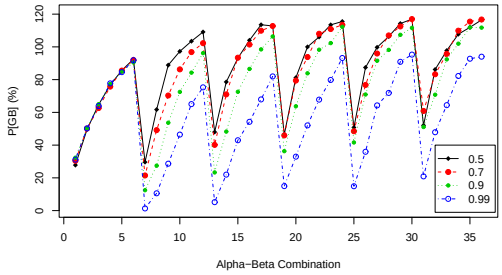
Figure 6.7 shows the results for the Talent Scheduling problem chosen to represent the group. In all three of the P_{GB} graphs as $\beta \rightarrow 5$, for all α values, the quality of the solutions found gets worse. For GBAS and MMAS, $\rho_{alg} = 0.99$ seems to fare the best, with AS preferring the slightly lower value of 0.9. Among the α values there is little difference in performance for Ant System, but with both GBAS and MMAS an increase in this parameter seems to degrade performance, the degree of which is dependent on ρ_{alg}. For instance, in Figure 6.7(e), for $\rho_{alg} = 0.5$ there is a clear decrease in solution quality with the increase in α. In contrast, the value of 0.99 seems to be affected little by changes in α.

As with the TSP problem, high ρ_{alg} for I_{GB} produces the latest convergence. The difference between these graphs and the TSP equivalents is that, for MMAS, where $\rho_{alg} < 0.99$ and $\alpha = 3$ very small values of I_{GB} are shown, in contrast to the TSP where this decline does not occur until $\alpha = 4$.

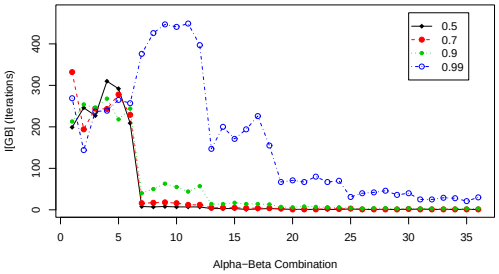
6.3.1.3 Static Random Cost Matrix

The Static Random Cost Matrix (SRCM) heuristic shows very similar characteristics to the RCM heuristic. For all three algorithms, Figure 6.8 shows a decrease in performance as β is increased for all α values. This is because there is no random chance that a bad cost on a good arc will improve given enough samples as with the dynamic heuristics.

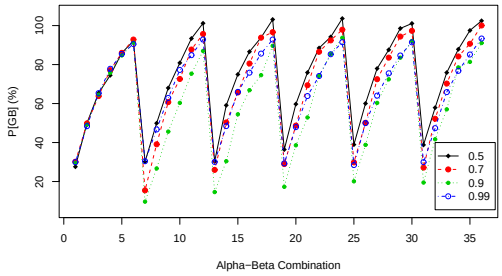
Figure 6.9 shows how the heuristic performs on a large Talent Scheduling problem. For all the algorithms, the performance is much flatter than for the TSP. The varying



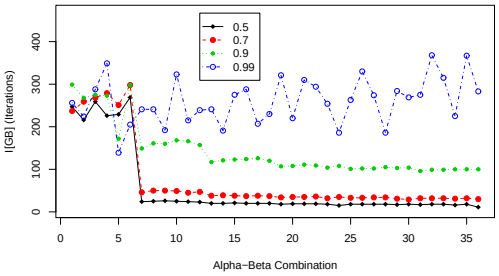
(a) P_{GB} using GBAS



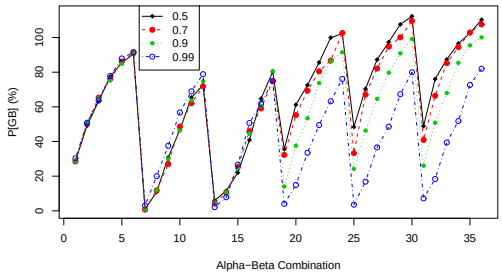
(b) I_{GB} using GBAS



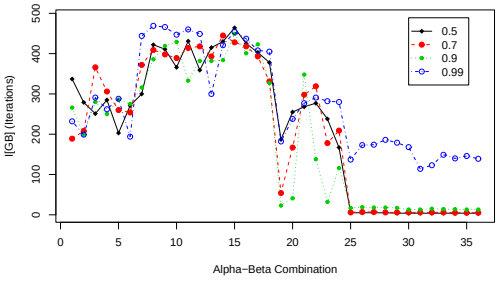
(c) P_{GB} using AS



(d) I_{GB} using AS

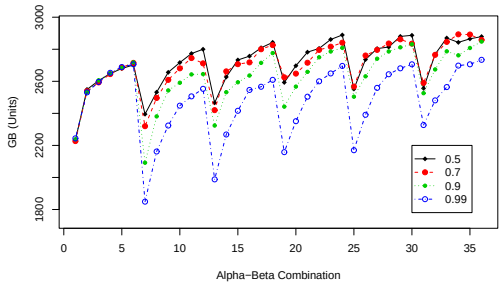


(e) P_{GB} using MMAS

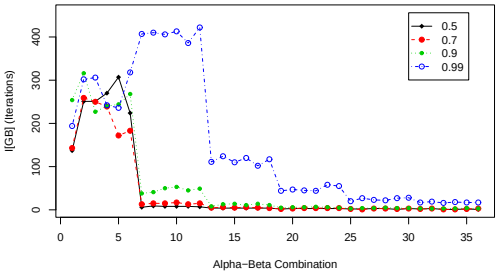


(f) I_{GB} using MMAS

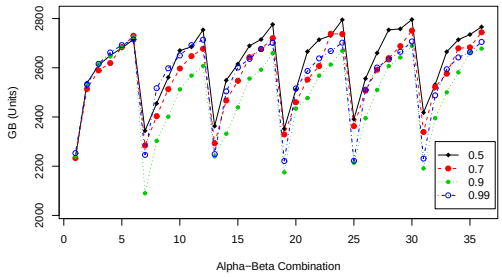
Figure 6.6: Example of the results achieved for the Reverse Cost Matrix (RCM) heuristic for the TSP domain, problem Ulysses16.



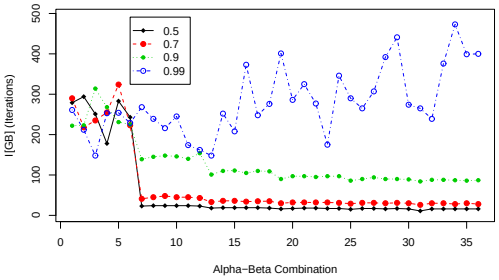
(a) P_{GB} using GBAS



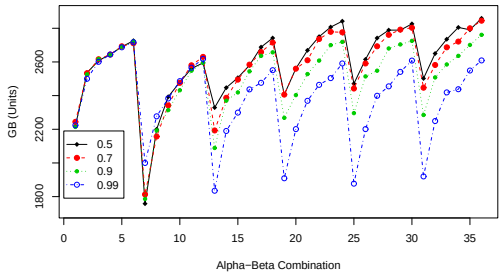
(b) I_{GB} using GBAS



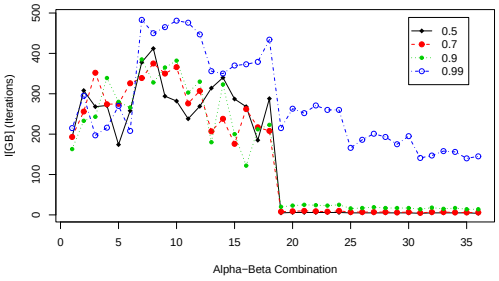
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 6.7: Example of the results achieved for the Reverse Cost Matrix (RCM) heuristic for the Talent Scheduling domain, problem talent_10_30_12.

of α and β has very little influence, producing an almost linear increase across values, unlike anything seen in earlier graphs. This cannot be explained by the associated I_{GB} graph because there does not appear to be a significant difference between the TSP and TS results.

The best solutions are achieved when $\alpha = 1, \beta = 3$ for GBAS, and $\beta = 5$ for Ant System. This is very unusual because the β value is so high compared to α, and may be a consequence of the problem not having a large range of costs available. The consequence of this is that the heuristic is more likely to be using a favourable cost matrix.

Ant System shows that $\rho_{alg} = 0.9$ performs the best of all the ρ_{alg} values, in contrary to the fact that the convergence is later for 0.99. This is an instance of where more iterations does not necessarily transform into better performance.

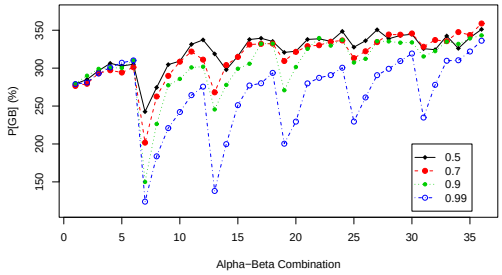
Finally MMAS has a very distinct pattern. For $\alpha = 1$, there is a fall in the objective value for all values of ρ_{alg} and β, this is accompanied at the same time by an increase in the convergence. Once $\alpha = 2$, both variables degrade significantly, only $\rho_{alg} = 0.99$ maintains similar performance levels throughout the variation in α and β.

6.3.1.4 Dynamic Random Increment

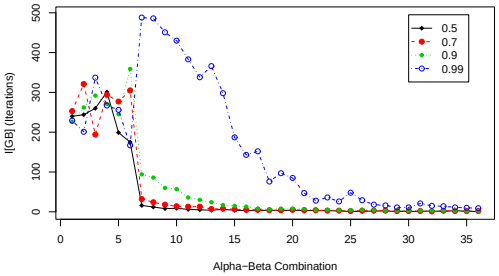
The Dynamic Random Increment heuristic (DRI) adds or subtracts a random amount from the heuristic on each call. It is expected that sometimes this value will be close to zero, and sometimes it will not be. Therefore, the expectation for this heuristic is that the Ant algorithms will sometimes be able to produce good solutions, thus the graphs will look more like those for the Nearest Neighbour heuristic.

In Figure 6.10 the results for the problem rd100 are shown. This TSP problem shows that the expected behaviour is produced from all three algorithms with the ρ_{alg} values bunched together. For MMAS, when $\rho_{alg} = 0.99$ and $\beta = 0$ the performance is not good, but then as β increases, so does the performance. In terms of convergence there is nothing unexpected, except that MMAS and AS, unlike in Chapter 5, seem to have good convergence for $\rho_{alg} = 0.99$.

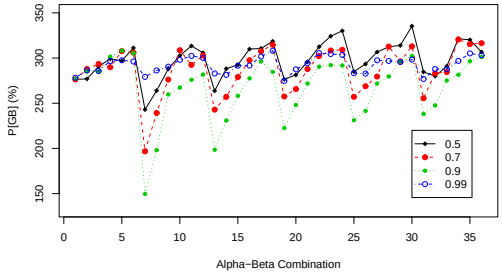
The Talent Scheduling domain is represented by Figure 6.11. This shows that all three algorithms have a preference for $\rho_{alg} = 0.99$, due to the late convergence that occurs for this setting. As α increases in GBAS and MMAS the performance degrades, although



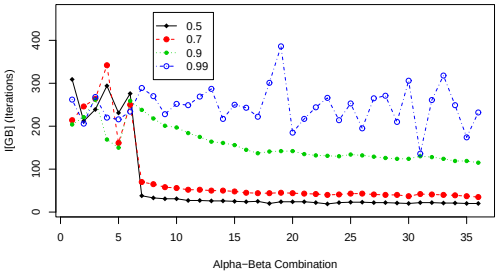
(a) P_{GB} using GBAS



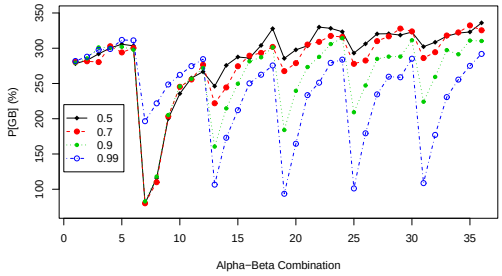
(b) I_{GB} using GBAS



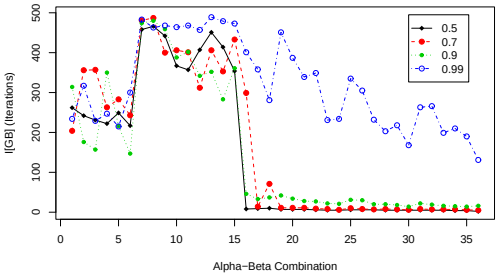
(c) P_{GB} using AS



(d) I_{GB} using AS

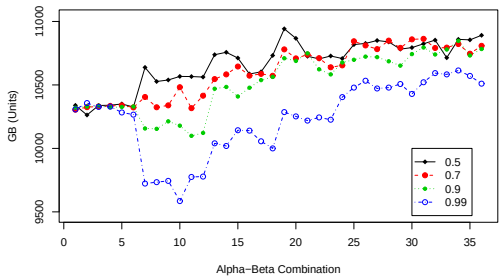


(e) P_{GB} using MMAS

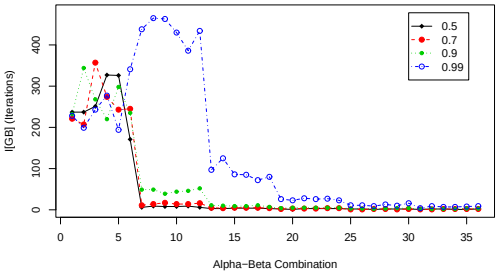


(f) I_{GB} using MMAS

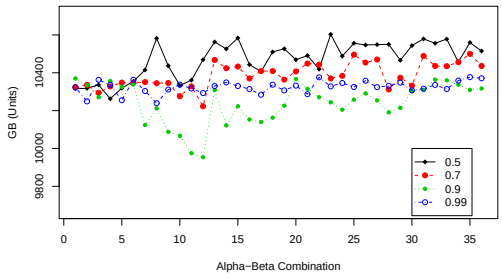
Figure 6.8: Example of the results achieved for the Static Random Cost Matrix (SRCM) heuristic for the TSP domain, problem Brazil58.



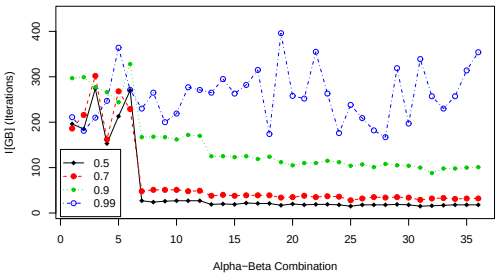
(a) P_{GB} using GBAS



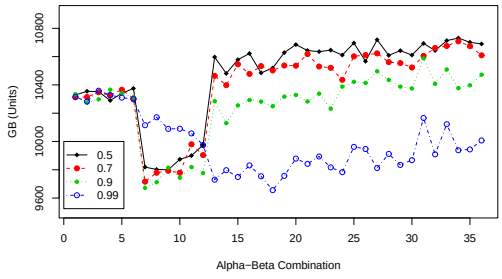
(b) I_{GB} using GBAS



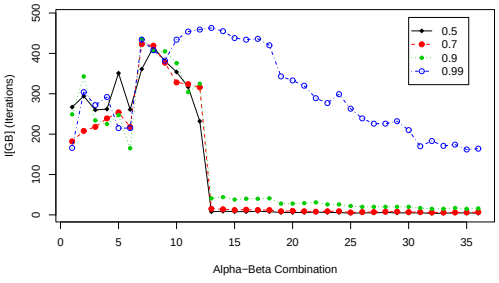
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 6.9: Example of the results achieved for the Static Random Cost Matrix (SRCM) heuristic for the Talent Scheduling domain, problem talent_10_100_2.

for AS there is little difference in performance. For all the algorithms, low values of ρ_{alg} result in better performance as β increases. In contrast, for higher ρ_{alg} values, as β increases the trend is to either stagnate or decay slightly.

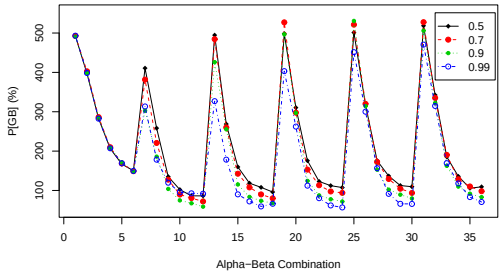
For ρ_{alg} values less than 0.99, at some point the convergence is barely above single figures, but this does not have an effect on the quality of solutions for the TSP. However, for the Talent Scheduling problem it results in a degrading of the solution quality for all the algorithms. Unlike in previous graphs for Ant System, the number of iterations before convergence occurs increases for $\rho_{alg} = 0.99$, as both α and β increase, perhaps due to extra exploration that this heuristic encourages. MMAS displays similar characteristics to the other heuristics with most of the search effort going on when α is small, and then a sharp fall once $\alpha = 3$.

6.3.1.5 Fixed Random Increment

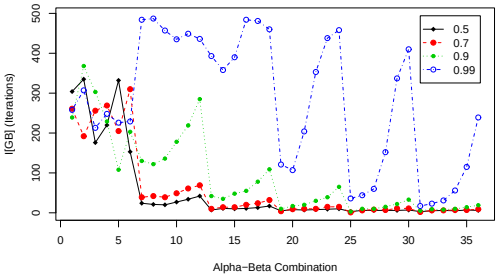
The Fixed Random Increment heuristic (FRI) on the TSP is illustrated by Figure 6.12. The results for this heuristic follow the pattern of the Nearest Neighbour heuristic. For MMAS, when $\alpha = 1$ the performance improves for $\rho_{alg} < 0.99$, but then the roles swap and $\rho_{alg} = 0.99$ is producing the best solutions, with performance decaying as ρ_{alg} decreases. When correlated with I_{GB} this behaviour is explained by $\rho_{alg} = 0.99$ maintaining a high number of iterations before convergence throughout the increase in α and β. In contrast, the lower values of ρ_{alg}, after $\alpha = 1$, start to move rapidly to convergence before 100 iterations.

The Ant System, for both P_{GB} and I_{GB}, provides standard looking graphs with $\rho_{alg} = 0.9$ performing the best over the range of α values, despite it being second latest to converge. GBAS does not have many distinguishing features either. However for $\rho_{alg} = 0.99$, for a given α, the increase in β has almost no effect on the convergence of the algorithm, thereby creating a step pattern in the graph.

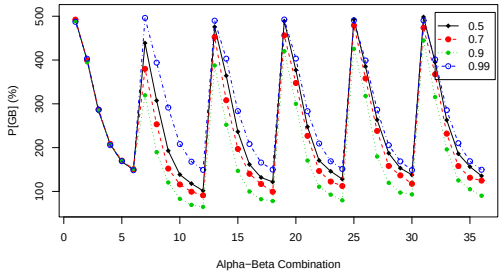
Figure 6.13 illustrates the results for Talent Scheduling. Again $\rho_{alg} = 0.99$ is the value that gives best performance for GBAS and MMAS, due to its convergence being the latest. For AS, $\rho_{alg} = 0.9$ has the second latest convergence, yet it still manages to get better results than $\rho_{alg} = 0.99$. Figure 6.13(e), illustrating the convergence characteristics of MMAS, has a distinct graph. In many of the α sections the performance improves up to $\beta = 2$, but then it decays. This is interesting as it happens even when $\alpha > \beta$. This behaviour does not seem to have encouragement from the convergence



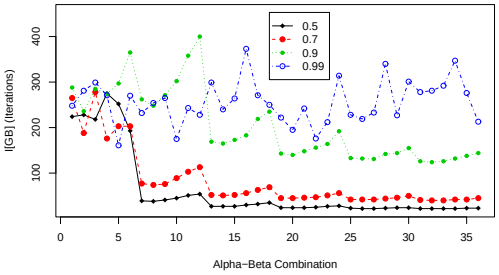
(a) P_{GB} using GBAS



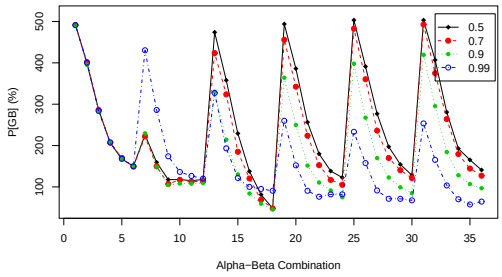
(b) I_{GB} using GBAS



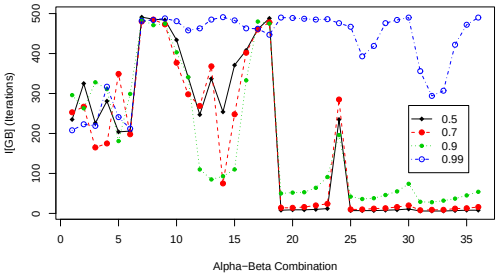
(c) P_{GB} using AS



(d) I_{GB} using AS

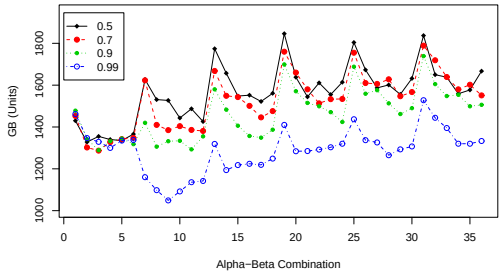


(e) P_{GB} using MMAS

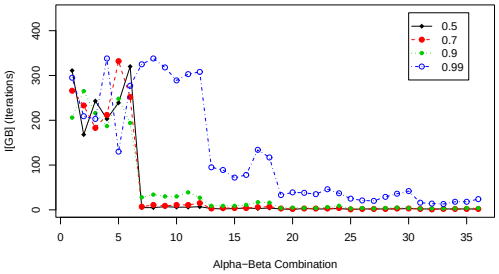


(f) I_{GB} using MMAS

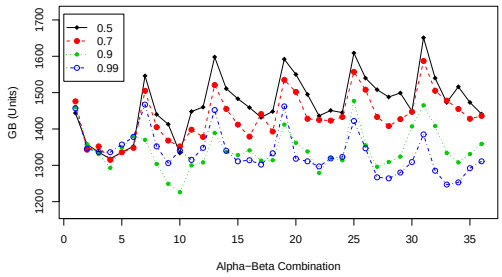
Figure 6.10: Example of the results achieved for the Dynamic Random Increment (DRI) heuristic for the TSP domain, problem rd100.



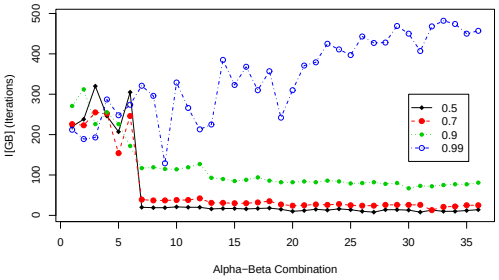
(a) P_{GB} using GBAS



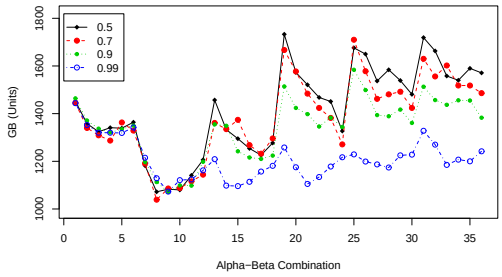
(b) I_{GB} using GBAS



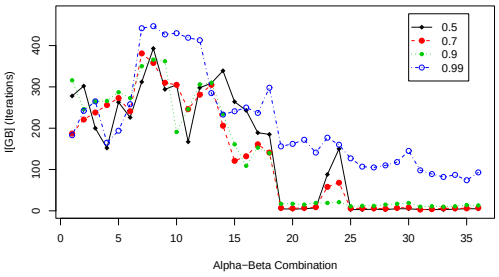
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 6.11: Example of the results achieved for the Dynamic Random Increment (DRI) heuristic for the Talent Scheduling domain, problem talent_10_20_6.

results, which appear similar to other heuristics.

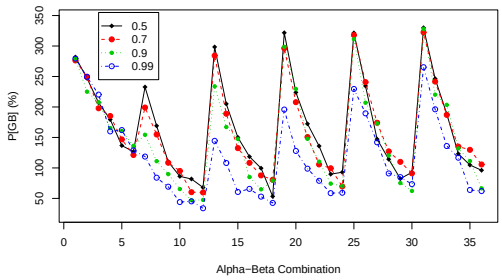
6.3.1.6 Comparison of all the heuristics

In Table 6.1 the performance of the various heuristics are compared to each other. To do this, for each of the 36 points, the median of the distribution was taken for each heuristic. These medians were then ranked from 1 to 6, for P_{GB}, 1 is the best and 6 is the worst, and for I_{GB}, it is the reverse. In the case of a tie, the points are given the same minimum value. For each point, the ranks of each heuristic are then summed, leaving six sums of ranks. These six totals are then ranked and the ordering is then displayed in this table. Each heuristic has been allocated a number to save space in the table, the assignment is as follows:

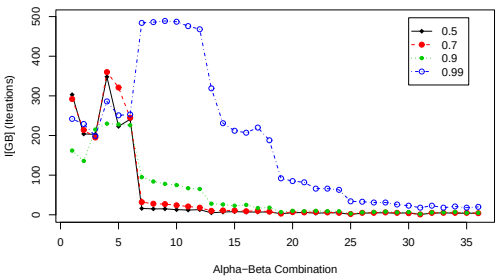
- 1 = NN,
- 2 = DRCM,
- 3 = RCM,
- 4 = SRCM,
- 5 = DRI,
- 6 = FRI.

What this table shows is the average relative performance with respect to the other heuristics for each ρ_{alg}. The expected ordering of the heuristics for P_{GB} was (NN, DRCM, DRI, SRCM, FRI, RCM), indicating that the Nearest Neighbour heuristic would be the best, then the dynamic heuristics and then finally the three misleading static heuristics. This ordering is shown in the table as $(1,2,5,4,6,3)$.

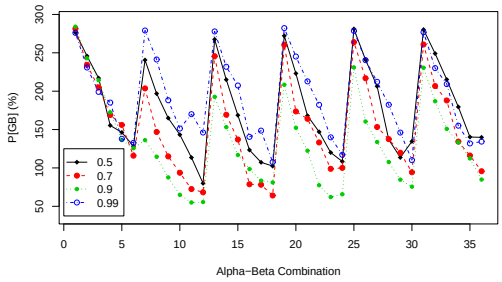
For I_{GB} the expected ordering was (NN, RCM, FRI, SRCM, DRI, DRCM) which is equivalent to $(1,3,6,4,5,2)$ in the numerical coding. The reason this ordering was expected is that the static heuristics are expected to converge faster than the runs using the dynamic heuristics. This is expected because the dynamic heuristics give fresh information that may suddenly produce a change in search direction. In contrast static heuristics offer no new information, so after a number of iterations the heuristic information results in little chance of a breakthrough. The misleading heuristics will lead to greater search, therefore they are expected to all rank above the Nearest Neighbour heuristic.



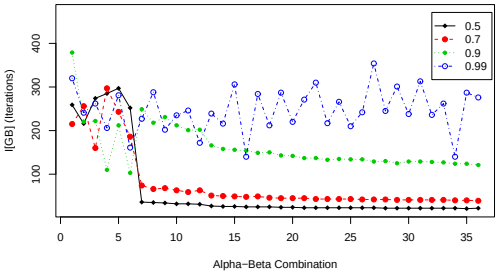
(a) P_{GB} using GBAS



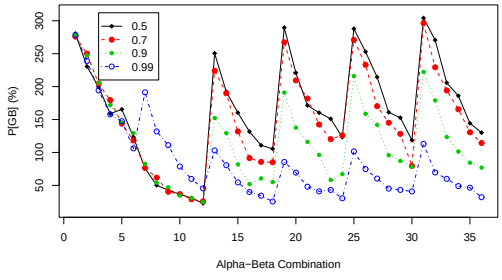
(b) I_{GB} using GBAS



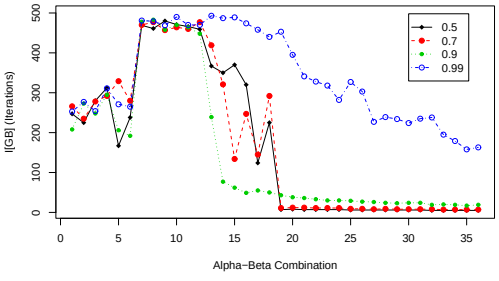
(c) P_{GB} using AS



(d) I_{GB} using AS

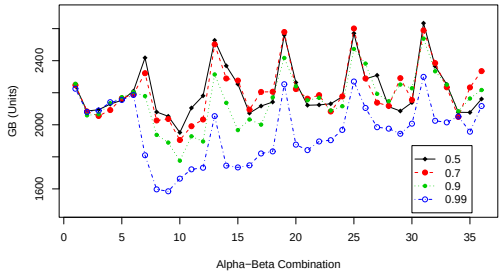


(e) P_{GB} using MMAS

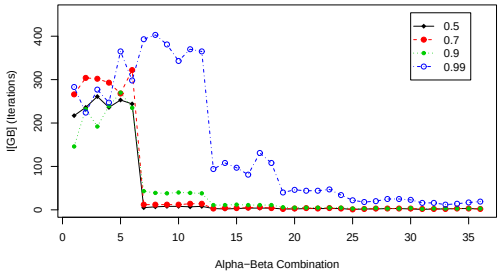


(f) I_{GB} using MMAS

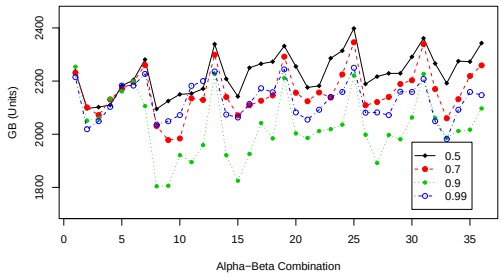
Figure 6.12: Example of the results achieved for the Fixed Random Increment (FRI) heuristic for the TSP domain, problem Brazil58.



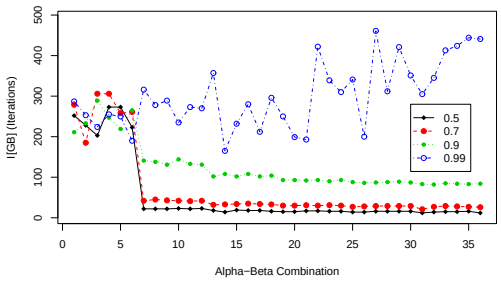
(a) P_{GB} using GBAS



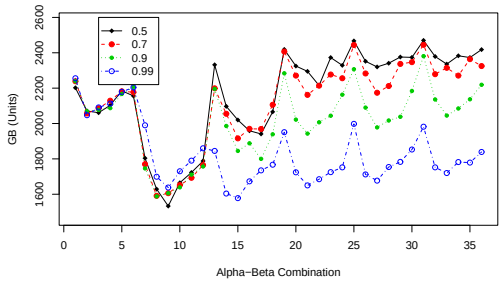
(b) I_{GB} using GBAS



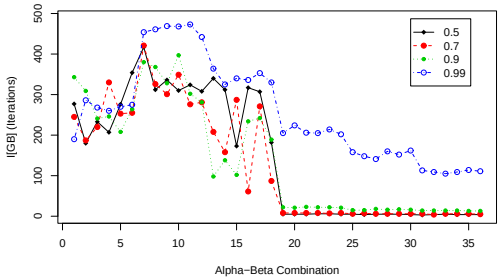
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 6.13: Example of the results achieved for the Fixed Random Increment (FRI) heuristic for the Talent Scheduling domain, problem talent_10_30_12.

Immediately one can see the expected orderings were in error. Most of the P_{GB} results ranked in the order of (NN, DRI, FRI, DRCM, SRCM, RCM). All the TSP problems showed this ordering indicating that a incremental change to the heuristic makes less difference than a random value. However, as expected the best was Nearest Neighbour heuristic, and the worst was the Random Cost Matrix heuristic.

For the Talent Scheduling, only 22% of the heuristics resulted in the same ordering as the TSP problem. The reason for this is the appropriateness of the Nearest Neighbour heuristic chosen for this domain. In the other orderings, 78% showed the best heuristics were DRI and FRI, with most of them in that order. This indicates that the increment may fix a deficiency in this heuristic.

The results for the variable I_{GB} (to the right of Table 6.1) are more complicated because for each ρ_{alg}, and each algorithm, there is a specific ordering. As expected the heuristics DRCM and DRI are the latest to converge, except for the occasional result where RCM is the last. From this, one can conclude that a heuristic can be used to direct the algorithm to new areas of search, aiding exploration as well as exploitation. In general, the two heuristics in the middle of the rankings are RCM and FRI, with NN and SRCM being the earliest to converge (all of which are supplied by a fixed cost matrix).

For Ant System, the dynamic heuristics that did lead to a later convergence when $\rho_{alg} < 0.99$, are beaten by the static heuristics as the best when $\rho_{alg} = 0.99$. This swap occurs because $\rho_{alg} = 0.99$ results in a significant increase in the number of iterations before convergence for the static heuristics, which is lost very quickly as ρ_{alg} is decreased.

Table 6.2 shows the ranks of the various ρ_{alg} values for each heuristic. As for the previous table, these results were calculated by summing the ranks of the medians of each distribution, at each of the 36 points. The ordering in the table is of $(0.5, 0.7, 0.9, 0.99)$, and for P_{GB}, 1 indicates the lowest solutions and therefore the best, and for I_{GB}, 4 indicates the latest convergence and therefore the best.

The expected ordering is $(0.5, 0.7, 0.9, 0.99)$ for GBAS, $(0.99, 0.5, 0.7, 0.9)$ for AS and MMAS, in accordance with Chapter 5. For GBAS, with the exception of a couple of entries, the observed results agree with only a few expectations.

For Ant System, the expected ordering was achieved only 19% of the time with the predominant orderings being $(0.5, 0.7, 0.99, 0.9)$ and $(0.5, 0.9, 0.99, 0.7)$ which occurred

Problem	P_{GB} GBAS	P_{GB} AS	P_{GB} MMAS	I_{GB} GBAS	I_{GB} AS	I_{GB} MMAS
$\rho_{alg} = 0.5$						
ulysses16	1,5,6,2,4,3	1,5,6,2,4,3	1,5,2,6,4,3	3,4,1,6,5,2	14,6,3,5,2	1,4,6,5,3,2
brazil58	1,5,6,2,4,3	1,5,6,2,4,3	1,5,6,2,4,3	3,4,1,6,5,2	1,4,6,3,5,2	4,1,63,5,2
rd100	1,5,6,2,4,3	1,5,6,2,4,3	1,5,6,2,4,3	3,4,1,6,2,5	1,4,6,3,5,2	4,1,3,6,5,2
talent_10_20_6	5,1,6,2,4,3	5,6,1,2,4,3	5,6,1,2,4,3	4,1,6,3,5,2	6,4,1,5,2,3	6,4,1,5,3,2
talent_10_30_12	5,6,1,2,4,3	5,1,6,2,4,3	5,6,1,2,4,3	4,6,1,3,5,2	4,6,1,5,2,3	4,6,1,3,5,2
talent_10_100_2	1,5,6,2,4,3	1,6,5,2,4,3	1,5,6,2,4,3	4,1,6,5,3,2	1,54,6,2,3	1,54,6,2,3
$\rho_{alg} = 0.7$						
ulysses16	1,5,6,2,4,3	1,5,6,2,4,3	1,5,2,6,4,3	4,1,3,6,5,2	1,4,6,3,5,2	1,4,6,5,3,2
brazil58	1,5,6,2,4,3	1,5,6,2,4,3	1,5,6,2,4,3	3,4,1,6,5,2	1,4,6,3,5,2	1,4,6,53,2
rd100	1,5,6,2,4,3	1,5,6,2,4,3	1,5,6,2,4,3	43,1,6,5,2	1,4,6,3,5,2	1,4,6,3,5,2
talent_10_20_6	5,6,1,2,4,3	6,5,1,2,4,3	5,1,6,2,4,3	6,1,4,5,3,2	4,6,5,1,2,3	6,4,1,5,3,2
talent_10_30_12	5,6,1,2,4,3	5,6,1,2,4,3	5,6,1,2,4,3	4,1,6,3,5,2	4,6,1,5,3,2	6,4,1,5,3,2
talent_10_100_2	5,1,6,2,4,3	1,6,5,4,2,3	1,6,5,2,4,3	4,6,1,5,3,2	4,1,6,5,2,3	4,1,6,5,3,2
$\rho_{alg} = 0.9$						
ulysses16	1,5,6,2,4,3	1,5,6,2,4,3	1,5,6,2,4,3	1,4,6,3,5,2	1,4,6,5,3,2	1,4,6,5,2,3
brazil58	1,5,6,2,4,3	1,5,6,2,4,3	1,5,6,2,4,3	4,1,3,6,5,2	1,4,6,3,5,2	1,4,6,5,2,3
rd100	1,5,6,2,4,3	1,5,6,2,4,3	1,5,6,2,4,3	1,4,3,6,5,2	1,4,6,3,5,2	1,4,6,3,5,2
talent_10_20_6	5,6,1,2,4,3	6,1,5,2,4,3	1,5,6,2,4,3	6,5,1,2,4,3	4,6,1,5,3,2	4,1,6,5,2,3
talent_10_30_12	5,6,1,2,4,3	5,1,6,2,4,3	5,1,6,2,4,3	4,6,1,3,5,2	4,6,1,5,3,2	4,6,1,3,5,2
talent_10_100_2	6,5,1,2,4,3	1,5,6,2,4,3	1,6,5,2,4,3	4,1,6,3,5,2	4,1,56,2,3	4,1,5,6,3,2
$\rho_{alg} = 0.99$						
ulysses16	1,5,6,2,4,3	1,5,6,2,4,3	1,5,6,2,4,3	1,4,6,53,2	2,43,1,5,6	1,6,4,5,23
brazil58	1,5,6,2,4,3	1,5,6,2,4,3	1,5,6,2,4,3	1,4,6,3,5,2	5,23,6,4,1	1,4,6,5,3,2
rd100	1,5,6,2,4,3	1,5,6,2,4,3	1,5,6,2,4,3	1,4,3,6,5,2	2,3,6,5,4,1	1,4,6,3,2,5
talent_10_20_6	5,1,6,2,4,3	6,1,5,4,2,3	6,5,1,2,4,3	4,6,1,5,3,2	2,5,43,1,6	4,1,6,5,3,2
talent_10_30_12	5,1,6,2,4,3	6,5,1,2,4,3	5,6,1,2,4,3	4,1,6,3,5,2	2,5,3,1,6,4	4,6,1,5,3,2
talent_10_100_2	1,5,6,2,4,3	1,6,5,2,4,3	1,5,6,2,4,3	4,1,5,6,3,2	3,5,4,1,6,2	4,1,6,5,2,3

Table 6.1: Table showing the rankings of each heuristic, for each algorithm, for all ρ_{alg} values. (Heuristic Legend: 1=NN,2=DRCM,3=RCM,4=SRCM,5=DRI,6=FRI. Two numbers side-by-side indicates they have the same ranking.)

38% and 31% respectively. The expected ordering occurred mostly for the two largest TSP problems, while the other orderings occurred for the Talent Scheduling problems. MMAS did not perform to expectations either with the majority of results behaving like GBAS, and none performing as expected.

The main reason for displaying these results is that they show that the claim that misleading heuristics benefit from lower ρ_{alg} values is not true. The reasoning behind the lower ρ_{alg} value is that it would enable the algorithm to remove bad relationships faster. The fact that for the worst heuristic, which is RCM, it is still best policy to pick a high ρ_{alg} shows that there is something different happening.

For the variable I_{GB} the majority of the results are identical. The expected ordering for all the algorithms was $(0.5, 0.7, 0.9, 0.99)$ where $\rho_{alg} = 0.99$ gives the latest convergence. In the majority of cases this is indeed the case. In all cases $\rho_{alg} = 0.99$ is either the best or best equal value.

6.3.1.7 Summary

Having described in detail the results of each particular heuristic, in this section a brief summary of the main points is given. First, it is important to try and classify which heuristics cause various problems for the algorithms. In terms of performance, RCM and SRCM show the most degradation as β increases across all α. This is because they are the most inflexible of the heuristics.

As expected, FRI and DRI, provided results that showed patterns most similar to the Nearest Neighbour heuristic. The reason for this is that the incremental change only affects the results when it changes the overall distribution of the probabilities in a significant way. This change requires there to be a variety of increments both additional and subtracted. If this does not happen then the distribution increases, or decreases, as a group and therefore when the probabilities are normalised these changes are subdued. The DRCM heuristic provided the most varied results, in some cases providing results of a similar pattern to NN, but in the majority of cases the results are a mixture providing unpredictable results.

For the variable I_{GB}, the main observation was that, for many of the heuristics, there was little change from the patterns of Nearest Neighbour heuristic. For GBAS and MMAS, the set of heuristics that caused the most variation were the two dynamic

	P_{GB}			I_{GB}		
Problem	**GBAS**	**AS**	**MMAS**	**GBAS**	**AS**	**MMAS**
NN						
ulysses16	4,3,2,1	4,3,1,2	4,3,2,1	2,1,3,4	1,2,3,4	1,2,3,4
brazil58	4,3,2,1	4,2,1,3	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
rd100	4,3,2,1	3,2,1,4	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_20_6	4,3,2,1	4,3,2,1	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_30_12	4,3,2,1	4,3,1,2	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_100_2	3,4,2,1	4,3,1,2	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
DRCM						
ulysses16	4,3,2,1	4,2,1,3	4,2,1,3	1,2,3,4	1,2,3,4	2,1,3,4
brazil58	4,3,1,2	3,2,1,4	4,2,1,3	1,2,3,4	1,2,3,3	1,2,3,4
rd100	4,3,1,2	3,2,1,4	4,2,1,3	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_20_6	4,3,2,1	4,3,1,2	4,3,2,1	1,2,3,4	1,2,3,4	1,3,2,4
talent_10_30_12	4,3,2,1	4,3,1,2	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_100_2	4,3,2,1	4,3,1,2	4,3,1,1	1,2,3,4	1,2,3,4	1,2,3,4
RCM						
ulysses16	4,3,2,1	4,3,1,2	4,3,1,1	1,2,3,4	1,2,3,4	1,2,2,4
brazil58	4,3,2,1	4,2,1,3	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
rd100	4,3,2,1	4,2,1,3	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_20_6	4,3,2,1	4,3,1,2	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_30_12	4,3,2,1	4,2,1,3	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_100_2	4,3,2,1	4,2,1,3	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
SRCM						
ulysses16	3,4,2,1	4,3,1,2	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
brazil58	4,3,2,1	4,2,1,3	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
rd100	4,3,2,1	4,2,1,3	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_20_6	4,3,2,1	4,3,2,1	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_30_12	4,3,2,1	4,3,1,2	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_100_2	4,3,2,1	4,3,1,2	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
DRI						
ulysses16	4,3,2,1	4,2,1,3	4,3,2,1	1,2,3,4	1,2,3,4	2,3,1,4
brazil58	4,3,2,1	3,2,1,4	4,3,1,2	1,2,3,4	1,2,3,4	1,2,3,4
rd100	4,3,2,1	3,2,1,4	4,3,1,2	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_20_6	4,3,2,1	4,3,2,1	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_30_12	4,3,2,1	4,2,1,3	4,3,2,1	1,2,3,4	1,2,3,4	2,1,3,4
talent_10_100_2	4,3,2,1	4,3,1,2	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
FRI						
ulysses16	4,3,2,1	4,2,1,3	4,3,2,1	1,2,3,4	1,2,3,4	1,1,3,4
brazil58	4,3,2,1	3,2,1,4	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
rd100	4,3,2,1	3,2,1,4	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_20_6	4,3,2,1	4,3,2,1	4,3,2,1	2,1,3,4	1,2,3,4	1,2,3,4
talent_10_30_12	4,3,2,1	4,3,1,2	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4
talent_10_100_2	4,3,2,1	4,3,1,2	4,3,2,1	1,2,3,4	1,2,3,4	1,2,3,4

Table 6.2: Table showing the ranks of each ρ_{alg} for each heuristic. The ranks are shown in the order $\{0.5, 0.7, 0.9, 0.99\}$.

heuristics, DRCM and DRI. DRCM also provided difficulties for Ant System. In general, these difficulties took the form of increases in the number of iterations it took to find the best solution for a particular run for $\rho_{alg} \in \{0.9, 0.99\}$. MMAS did not do well if α went above 2 or 3, depending on the heuristic, at which point the values crashed to less than 100 iterations. This was not only for the misleading heuristics, but this also happened for Nearest Neighbour heuristic when $\alpha > 2$. The explanation for this is unclear but it is most likely an aspect of the pheromone matrix limits. For the other algorithms, the performance is degraded, but they still follow similar trends to the Nearest Neighbour heuristic.

The setting $\rho_{alg} = 0.99$ produced the best convergence for all heuristics, and for all algorithms. However, for Ant System this is not transfered to creating better quality solutions, thus the best ρ_{alg} value is 0.9. The inability for Ant System to convert greater search to better quality solutions is most probably linked to the unbounded nature of the values in the pheromone matrix.

In the final part of this subsection the heuristics were ordered in terms of performance and the most common ordering was (NN,DRI,FRI,DRCM,FRCM,RCM). It showed that small errors in the heuristic did not make that much of a difference compared to the Nearest Neighbour heuristic, but that fixed errors for an entire distribution of arcs did cause serious performance problems. In terms of I_{GB}, the ranking is less distinct, but can be put into this list ({NN,SRCM},{RCM,FRI},{DRCM,DRI}), where those between curly brackets are found in either order. The pattern suggests that the dynamic heuristics provide more time to search before converging than the other fixed heuristics. This is a property that will be investigated further in Subsection 6.3.3.

6.3.2 Comparison to Nearest Neighbour Heuristic

Besides studying each heuristic's results separately, it is important to put them into context. Therefore in this section each heuristic is compared with the Nearest Neighbour heuristic. This is achieved by discussing figures where the medians of each heuristic are subtracted from each other. Therefore negative values indicate where the Nearest Neighbour heuristic was beaten, either in solution quality or in convergence.

6.3.2.1 DRCM compared to NN

Figure 6.14 illustrates the results achieved when comparing the Dynamic Random Cost Matrix heuristic (DRCM) to the Nearest Neighbour heuristic (NN). For all three algorithms, $\rho_{alg} = 0.99$ shows the greatest decrease in performance. For the middle values of ρ_{alg} there are small improvements as β is increased, but they are small. On average the solution quality decays by approximately 250-300%.

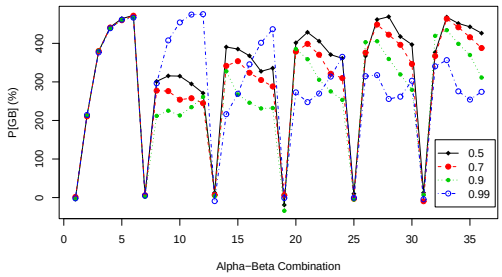
The convergence is delayed by approximately 400 iterations more for the high ρ_{alg} values, than for the Nearest Neighbour heuristic. This demonstrates that this dynamic heuristic allows some extra search to occur. For Ant System, the difference in I_{GB} is better for $\rho_{alg} = 0.9$, but worse than the Nearest Neighbour heuristic for $\rho_{alg} = 0.99$.

For the Talent Scheduling problems, illustrated in Figure 6.15, the performance of both variables is similar to the TSP. In general, the values of I_{GB} for the GBAS and AS algorithms show less of a difference for most of the ρ_{alg} values; only 0.99 produces any major differences for the benefit of DRCM in the case of GBAS and the benefit of NN in the case of Ant System.

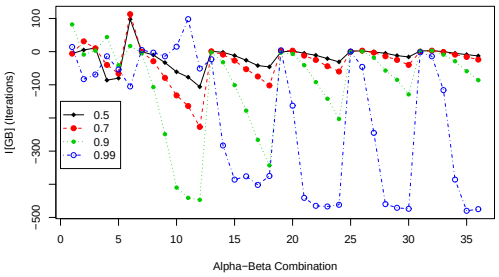
To solidify the discussion of the figures the following hypothesis has been constructed:

Null Hypothesis:	There is no significant difference between DRCM and NN when using the GBAS algorithm for P_{GB} for TSP.
Alternative Hypothesis:	DRCM is significantly greater than NN for the variable P_{GB} when using the GBAS algorithm for TSP.

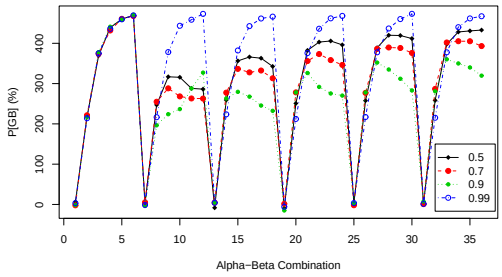
This hypothesis is repeated for AS and MMAS instead of GBAS, I_{GB} instead of P_{GB} and Talent Scheduling (TS) for TSP. For I_{GB}, the Alternative Hypothesis is a one-tailed test, but it represents better convergence characteristics. For P_{GB}, this hypothesis represents a worse characteristic. This gives a total of twelve hypotheses to test.



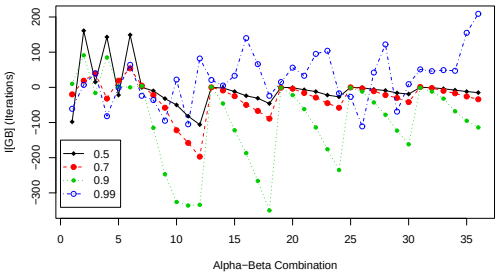
(a) P_{GB} using GBAS



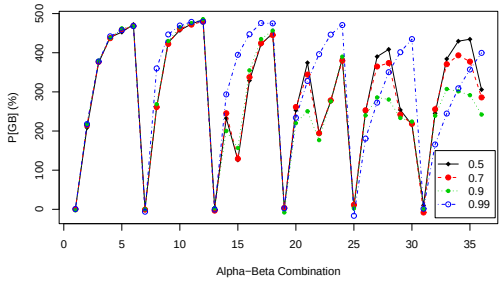
(b) I_{GB} using GBAS



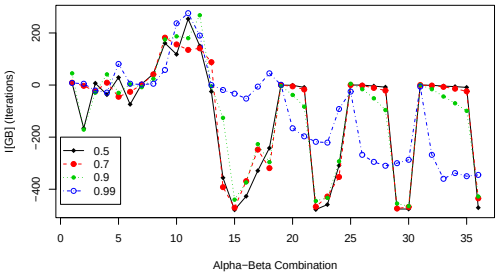
(c) P_{GB} using AS



(d) I_{GB} using AS

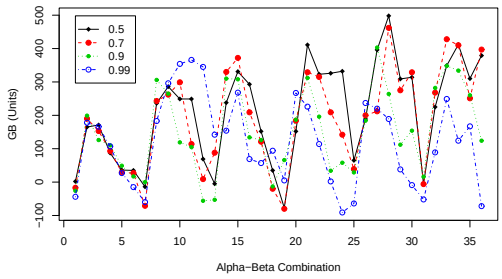


(e) P_{GB} using MMAS

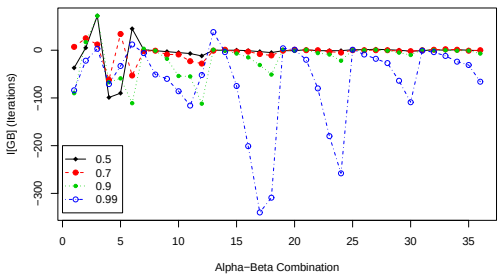


(f) I_{GB} using MMAS

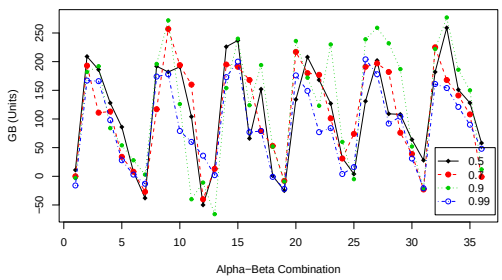
Figure 6.14: Figures showing the difference of the DRCM heuristic from the Nearest Neighbour heuristic for the TSP domain, problem rd100. (Negative values indicate that DRCM performed better for that variable than NN.)



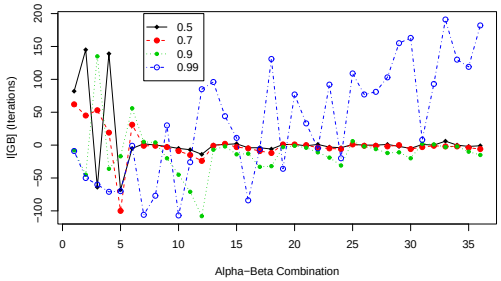
(a) P_{GB} using GBAS



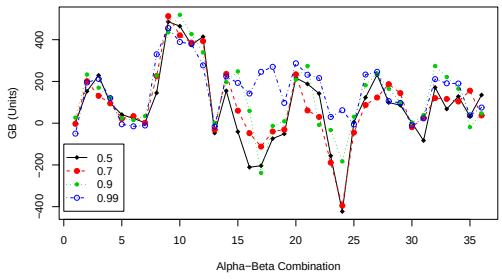
(b) I_{GB} using GBAS



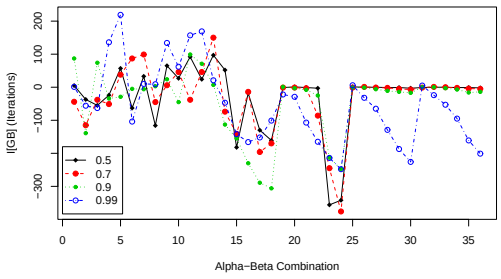
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 6.15: Figures illustrating the difference between the DRCM heuristic and the Nearest Neighbour heuristic for the Talent Scheduling domain, problem talent_10_30_12. (Negative values indicate that DRCM performed better for that variable than NN.)

Problem	ρ_{alg}	P_{GB}			I_GB		
		GBAS	AS	MMAS	GBAS	AS	MMAS
ulysses16	0.5	$< 0.0001^*$	$< 0.0001^*$	0.0068^{*B}	0.0045^{*B}	$< 0.0001^*$	$< 0.0001^*$
ulysses16	0.7	$< 0.0001^*$	$< 0.0001^*$	0.0035^*	0.0001^*	$< 0.0001^*$	$< 0.0001^*$
ulysses16	0.9	$< 0.0001^*$	$< 0.0001^*$	0.0003^*	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$
ulysses16	0.99	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.9615	$< 0.0001^*$
brazil58	0.5	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.0010^*	$< 0.0001^*$	0.0001^*
brazil58	0.7	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.0021^*	$< 0.0001^*$	0.0001^*
brazil58	0.9	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$
brazil58	0.99	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.7153	$< 0.0001^*$
rd100	0.5	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.0030^*	0.0001^*	0.0004^*
rd100	0.7	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.0024^*	$< 0.0001^*$	0.0004^*
rd100	0.9	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.0001^*	$< 0.0001^*$	0.0001^*
rd100	0.99	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.9642	0.0006^*
talent_10_20_6	0.5	0.0026^*	$< 0.0001^*$	0.1721	0.2343	0.0745	0.0936
talent_10_20_6	0.7	0.0409^{*B}	$< 0.0001^*$	0.1087	0.1102	0.1510	0.0079^{*B}
talent_10_20_6	0.9	0.2604	$< 0.0001^*$	0.0202^{*B}	0.0167	0.0204^{*B}	0.0403^{*B}
talent_10_20_6	0.99	0.1991	$< 0.0001^*$	$< 0.0001^*$	0.0075^{*B}	0.9994	0.0062^{*B}
talent_10_30_12	0.5	$< 0.0001^*$	$< 0.0001^*$	0.0910	0.3018	0.2951	0.1097
talent_10_30_12	0.7	$< 0.0001^*$	$< 0.0001^*$	0.0723	0.2453	0.0610	0.0378^{*B}
talent_10_30_12	0.9	0.0001^*	$< 0.0001^*$	0.0009^*	0.0929	0.0627	0.0357^{*B}
talent_10_30_12	0.99	0.0046^{*B}	$< 0.0001^*$	$< 0.0001^*$	0.0151^{*B}	0.9925	0.0274^{*B}
talent_10_100_2	0.5	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.5764	0.4305	0.0836
talent_10_100_2	0.7	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.5045	0.2182	0.0667
talent_10_100_2	0.9	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.2036	0.1151	0.0649
talent_10_100_2	0.99	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	0.0253^{*B}	0.2442	0.0320^{*B}

Table 6.3: Table showing the p-values for testing whether the DRCM heuristic was worse than the NN heuristic. (A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\frac{0.05}{12} = 0.0042$.)

From Table 6.3 one can conclude the following:

- For P_{GB} and GBAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and AS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and MMAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and GBAS on TS, the Null Hypothesis can be rejected.
- For P_{GB} and AS on TS, the Null Hypothesis can be rejected.
- For P_{GB} and MMAS on TS, the Null Hypothesis can be rejected.
- For I_{GB} and GBAS on TSP, the Null Hypothesis can be rejected.
- For I_{GB} and AS on TSP, the Null Hypothesis can be rejected.
- For I_{GB} and MMAS on TSP, the Null Hypothesis can be rejected.
- For I_{GB} and GBAS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and AS on TS, the Alternative Hypothesis can be rejected.

- For I_{GB} and MMAS on TS, the Null Hypothesis can be rejected, although most of the largest problems returned insignificant differences.

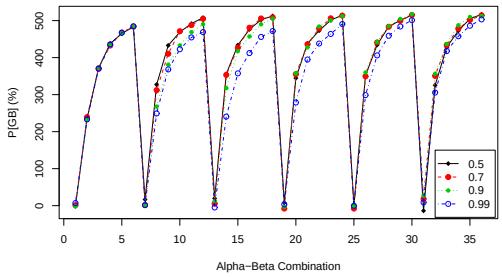
For the DRCM heuristic, where random values are generated for each heuristic call in the range $[1, c_{max}]$, there is a significant performance decrease for all three of the algorithms. In terms of convergence, the heuristic allows the algorithm to search more of the space for the TSP. However for the Talent Scheduling problems, there is no difference in the convergence characteristic on most of the problems for GBAS and Ant System. For MMAS, there was a mixed result with the marginal majority showing a significant difference.

6.3.2.2 RCM compared to NN

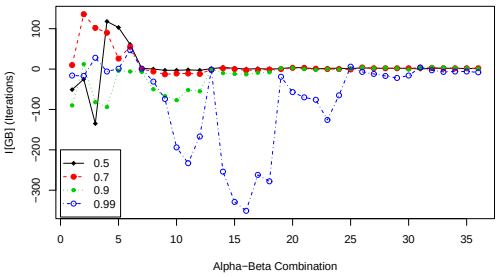
The Reverse Cost Matrix (RCM), when compared to the the Nearest Neighbour heuristic, has been shown to be the worst of all the heuristics. This is because it is intended to constantly convey the worst information to the algorithm. In terms of solution quality all the algorithms shown in Figure 6.16 perform similarly badly, with RCM being approximately 500% worse than the solutions generated by the Nearest Neighbour heuristic. As β is increased for GBAS and MMAS, $\rho_{alg} = 0.99$ sometimes breaks away, and depending on α may improve or degrade the quality of the solutions.

The convergence for all three algorithms is very similar, with all ρ_{alg} not straying too far from the values the Nearest Neighbour heuristic returned. For GBAS, $\rho_{alg} = 0.99$ enabled more search that the other values, explaining the slight difference in P_{GB}. However, Ant System was more varied with $\rho_{alg} = 0.99$ both delaying, and also causing early convergence. There seems to be no relationship between the α, β-pair and when the changes in convergence occur. However, a very general rule is that when $\beta < \alpha$ the convergence is delayed and vice versa. In contrast, MMAS is very similar to GBAS, except for when $\alpha = 2$ at which point the search increases for all ρ_{alg}, and then as α continues to increase, only $\rho_{alg} = 0.99$ continues to deviate from the Nearest Neighbour heuristic.

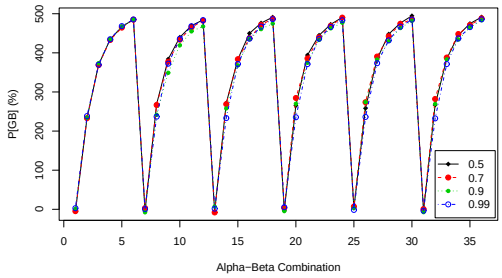
The Talent Scheduling problem, in Figure 6.17, shows a similar pattern to the TSP graphs. All three algorithms result in poor quality solutions. Furthermore, the convergence graphs show very little difference between the two heuristics for all the algorithms, in a similar manner to the TSP. However, the exception is the trough when $\alpha = 2$ in Figure 6.17(f), which is must less pronounced than for the TSP.



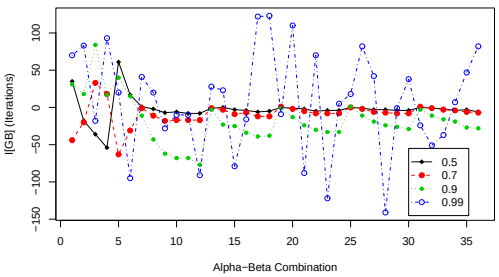
(a) P_{GB} using GBAS



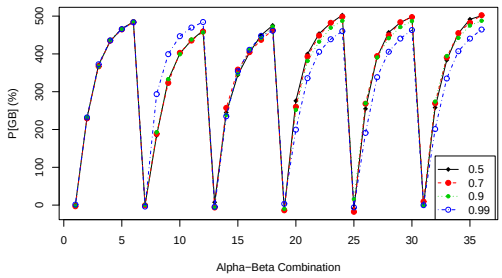
(b) I_{GB} using GBAS



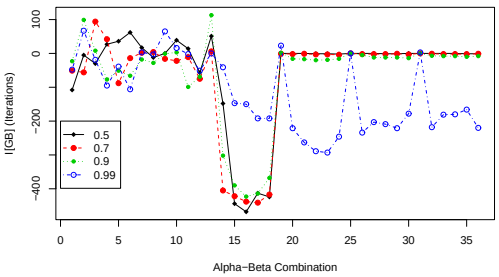
(c) P_{GB} using AS



(d) I_{GB} using AS

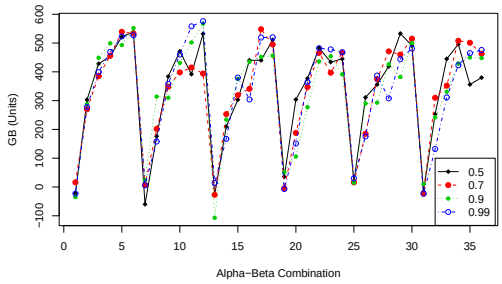


(e) P_{GB} using MMAS

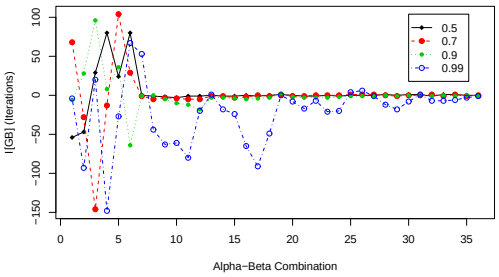


(f) I_{GB} using MMAS

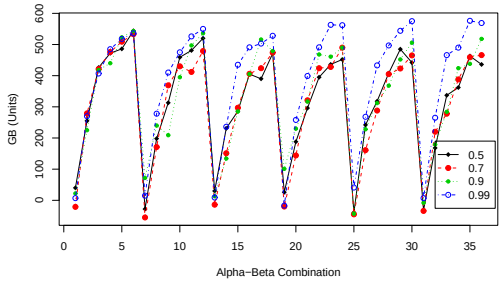
Figure 6.16: Figures showing the difference of the RCM heuristic from the NN heuristic for the TSP domain, problem Brazil58. (Negative values indicate that RCM performed better for that variable than NN.)



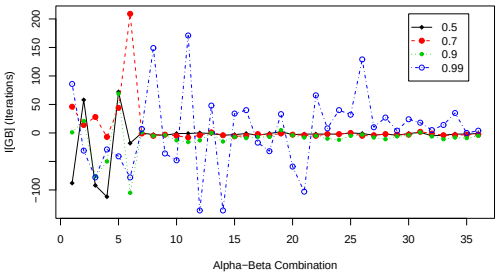
(a) P_{GB} using GBAS



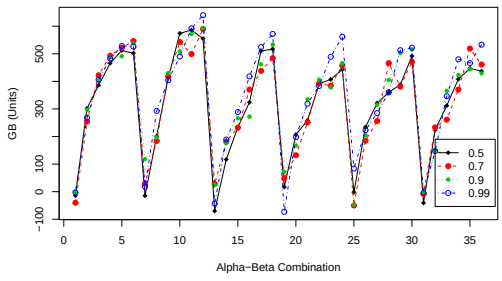
(b) I_{GB} using GBAS



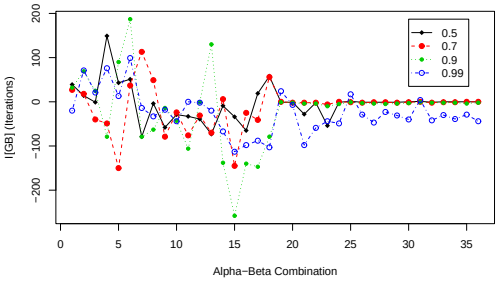
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 6.17: Figures illustrating the difference between the RCM heuristic and the NN heuristic for the Talent Scheduling domain, problem talent_10_20_6. (Negative values indicate that RCM performed better for that variable than NN.)

Problem	ρ_{alg}	P_{GB}			I_GB		
		GBAS	AS	MMAS	GBAS	AS	MMAS
ulysses16	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.9746	0.0276^{*B}	0.0121^{*B}
ulysses16	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.7189	0.0119^{*B}	0.0050^{*B}
ulysses16	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.1211	0.0006*	0.0022*
ulysses16	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.0056^{*B}	0.7848	< 0.0001*
brazil58	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.9781	0.0091^{*B}	0.1536
brazil58	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.8920	0.0062^{*B}	0.0223^{*B}
brazil58	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.6218	0.0024*	0.0091^{*B}
brazil58	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.0164^{*B}	0.8564	< 0.0001*
rd100	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.9394	0.0224^{*B}	0.1737
rd100	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.7603	0.0158^{*B}	0.0759
rd100	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.6412	0.0043^{*B}	0.0323^{*B}
rd100	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.0649	0.9187	0.0146^{*B}
talent_10_20_6	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.4633	0.0110^{*B}	0.3046
talent_10_20_6	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.2391	0.0189^{*B}	0.0937
talent_10_20_6	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.1024	0.0434	0.1473
talent_10_20_6	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.1410	0.6819	0.1324
talent_10_30_12	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.3423	0.0948	0.3086
talent_10_30_12	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.3367	0.1126	0.3014
talent_10_30_12	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.2760	0.2476	0.2567
talent_10_30_12	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.2103	0.7322	0.2071
talent_10_100_2	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.4593	0.0311	0.0476^{*B}
talent_10_100_2	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.3992	0.0499	0.2016
talent_10_100_2	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.4019	0.1005	0.2406
talent_10_100_2	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.1706	0.7662	0.0461^{*B}

Table 6.4: Table showing the p-values for testing whether the RCM heuristic was worse than the NN heuristic. (A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\frac{0.05}{12} = 0.0042$.)

As for the previous heuristic there are 12 hypotheses to be tested. In general, the expectation is that the performance will be worse for the variable P_{GB}, but this time there will be a very small difference for the I_{GB}. However, if there is any difference, RCM will be greater than the Nearest Neighbour heuristic.

From table 6.4 one can conclude the following:

- For P_{GB} and GBAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and AS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and MMAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and GBAS on TS, the Null Hypothesis can be rejected.
- For P_{GB} and AS on TS, the Null Hypothesis can be rejected.
- For P_{GB} and MMAS on TS, the Null Hypothesis can be rejected.
- For I_{GB} and GBAS on TSP, the Alternative Hypothesis can be rejected.
- For I_{GB} and AS on TSP, the Null Hypothesis can be rejected, except for $\rho_{alg} = 0.99$ where the Alternative Hypothesis should be rejected.

- For I_{GB} and MMAS on TSP, the Null Hypothesis can be rejected, as n gets larger this is less significant.
- For I_{GB} and GBAS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and AS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and MMAS on TS, the Alternative Hypothesis can be rejected.

For P_{GB}, the RCM heuristic significantly affects the performance of the algorithm. For I_{GB}, there is little difference; except for Ant System when $\rho_{alg} < 0.99$, and MMAS on small TSP problems.

6.3.2.3 SRCM compared to NN

Figures 6.18 and 6.19 illustrate the results for the SRCM heuristic, which is a static cost matrix filled with random values. The former graph, representing the TSP problem, shows similar characteristics to RCM. For all three algorithms, as β increases, performance degrades for all $\rho_{alg} < 0.99$. For GBAS and MMAS, $\rho_{alg} = 0.99$ is the best of these values, while Ant System shows little difference with the lower ρ_{alg} values.

Figure 6.18(b), illustrating the convergence of GBAS, shows that $\rho_{alg} = 0.99$ deviates the most, delaying convergence when $\alpha = 1$ for as much as 200 iterations. Then as α is increased, this difference is reduced. For Ant System, as for RCM, $\rho_{alg} = 0.99$ zigzags across the rest of the ρ_{alg} results, sometimes delaying convergence, while at other times reducing the convergence by as much as 100 iterations from the levels achieved by the Nearest Neighbour heuristic. This does not result in similar behaviour for P_{GB}, which instead shows a fairly steady incline throughout increases in β. MMAS is the most diverse, with $\alpha < 4$ producing better convergence for all values of ρ_{alg}, maximised at $\alpha = 1$ for $\rho_{alg} = 0.5$ and $\alpha = 2$ for $\rho_{alg} = 0.9$. The change can be as much as 200 iterations for the better, but also as α rises, for $\rho_{alg} = 0.5$ the convergence can be reduced by 100 iterations.

Figure 6.19 shows the results for a Talent Scheduling problem, and it demonstrates very similar results to the previous set of graphs. The performance for GBAS, when $\rho_{alg} = 0.99$, is no longer fairing so well, and in the majority of cases all the ρ_{alg} lines are clustered together. Correspondingly for I_{GB} there is almost no divergence. For Ant System, the solution quality when $\rho_{alg} = 0.9$ becomes slightly worse than for the other

Problem	ρ_{alg}	P_{GB}			I_GB		
		GBAS	AS	MMAS	GBAS	AS	MMAS
ulysses16	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.9339	0.5023	0.3751
ulysses16	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.8633	0.3717	0.2435
ulysses16	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.4414	0.1691	0.2037
ulysses16	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.1677	0.8782	$0.0074^{\star B}$
brazil58	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.9393	0.1340	0.5787
brazil58	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.7950	0.1193	0.2861
brazil58	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.6662	0.0919	0.2424
brazil58	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.3509	0.7798	$0.0281^{\star B}$
rd100	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.7237	0.1166	0.5786
rd100	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.6709	0.1034	0.4306
rd100	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.5180	0.0938	0.3141
rd100	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.3870	0.6899	0.0892
talent_10_20_6	0.5	< 0.0001*	< 0.0001*	0.0016*	0.8357	0.6421	0.5841
talent_10_20_6	0.7	< 0.0001*	< 0.0001*	0.0012*	0.5366	0.8243	0.5543
talent_10_20_6	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.6405	0.7341	0.7214
talent_10_20_6	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.7558	0.7304	0.7432
talent_10_30_12	0.5	< 0.0001*	< 0.0001*	0.0017*	0.8228	0.8881	0.8687
talent_10_30_12	0.7	< 0.0001*	< 0.0001*	0.0003*	0.8026	0.8138	0.7427
talent_10_30_12	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.8184	0.8408	0.7898
talent_10_30_12	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.7304	0.1230	0.8758
talent_10_100_2	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.9068	0.4392	0.3434
talent_10_100_2	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.8516	0.4910	0.5829
talent_10_100_2	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.7196	0.5202	0.5826
talent_10_100_2	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.5957	0.3551	0.5869

Table 6.5: Table showing the p-values for testing whether the SRCM heuristic was worse than the NN heuristic. (A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\frac{0.05}{12} = 0.0042$.)

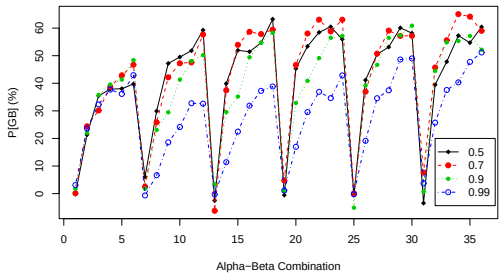
values, but the same degrading trend in solution quality is seen, as it was for GBAS. The associated graph for the variable I_{GB} shows the same varied results for $\rho_{alg} = 0.99$, but with the others very close together at zero.

MMAS, for the variable P_{GB}, when $\beta > 1$, favours $\rho_{alg} < 0.99$, in contrast to the TSP graph where $\rho_{alg} = 0.99$ is the best performing value. Figure 6.19(f) shows that for the Talent Scheduling problem there is less variation for $\alpha < 3$ than for the TSP, this explains the bunching of the results for the relative performance graphs.

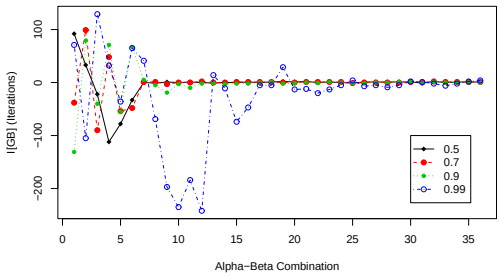
As for the previous heuristic there are 12 hypotheses to be tested. In general, the expectation is that the performance will be worse for the variable P_{GB}, but this time there will be little difference for the variable I_{GB}. As for the RCM heuristic, this heuristic will show a distribution greater than that of the Nearest Neighbour heuristic, therefore displaying a later convergence, if there is a difference in the two distributions.

From Table 6.5 one can conclude the following:

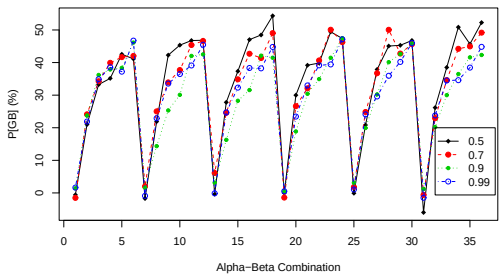
- For P_{GB} and GBAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and AS on TSP, the Null Hypothesis can be rejected.



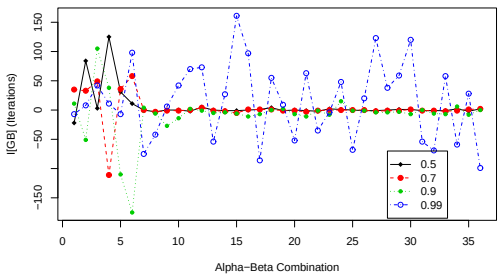
(a) P_{GB} using GBAS



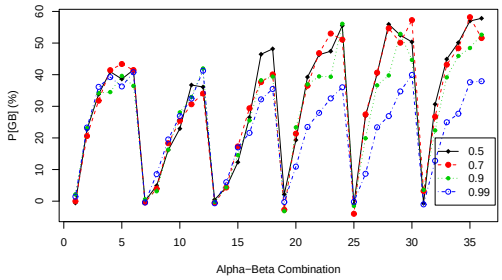
(b) I_{GB} using GBAS



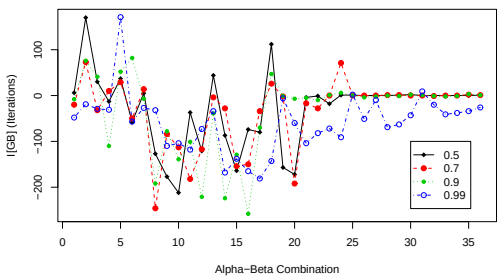
(c) P_{GB} using AS



(d) I_{GB} using AS

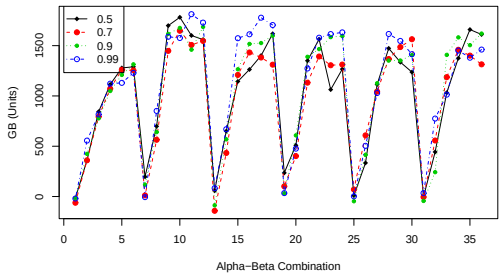


(e) P_{GB} using MMAS

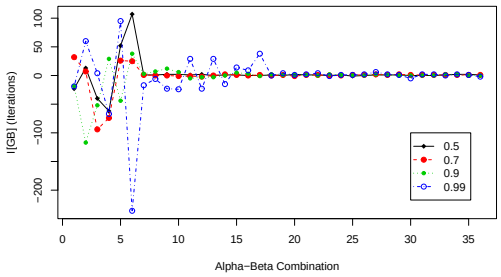


(f) I_{GB} using MMAS

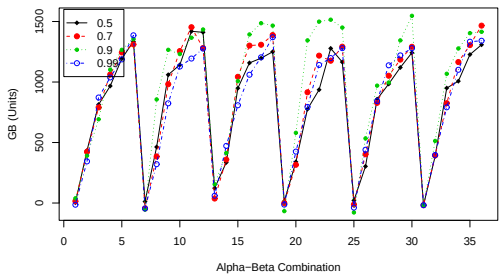
Figure 6.18: Figures illustrating the difference between the SRCM heuristic and the NN heuristic for the Talent Scheduling domain, problem Ulysses16. (Negative values indicate that SRCM performed better for that variable than NN.)



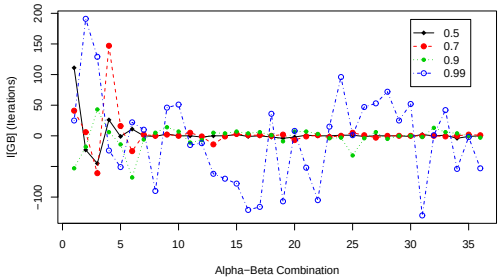
(a) P_{GB} using GBAS



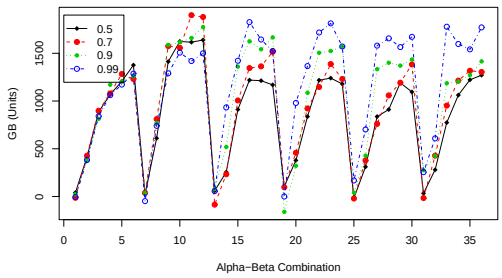
(b) I_{GB} using GBAS



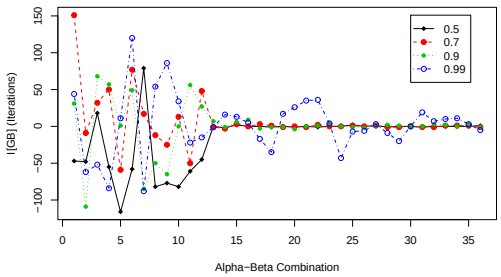
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 6.19: Figures illustrating the difference between the SRCM heuristic and the NN heuristic for the Talent Scheduling domain, problem talent_10_100_2. (Negative values indicate that SRCM performed better for that variable than NN.)

- For P_{GB} and MMAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and GBAS on TS, the Null Hypothesis can be rejected.
- For P_{GB} and AS on TS, the Null Hypothesis can be rejected.
- For P_{GB} and MMAS on TS, the Null Hypothesis can be rejected.
- For I_{GB} and GBAS on TSP, the Alternative Hypothesis can be rejected.
- For I_{GB} and AS on TSP, the Alternative Hypothesis can be rejected.
- For I_{GB} and MMAS on TSP, the Alternative Hypothesis can be rejected.
- For I_{GB} and GBAS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and AS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and MMAS on TS, the Alternative Hypothesis can be rejected.

Therefore, given these tests, there are two very clear conclusions from these results. The first conclusion is that SRCM offers significantly worse performance than NN. The second conclusion is that the SRCM heuristic does not alter the convergence characteristic of the algorithm significantly.

6.3.2.4 DRI compared to NN

Figures 6.20 and 6.21 show graphs representing the results for the Dynamic Random Increment heuristic (DRI), which adds or subtracts, with an equal probability, a random amount to the original heuristic value of Nearest Neighbour heuristic. The first observation is that these graphs are different to those seen previously. There are elements of both good and bad heuristic characteristics in these graphs.

In Figure 6.20 for the TSP, all three P_{GB} graphs show that the performance of DRI is worse than NN by approximately 90-150%. However, for each α, as β is increased the solution quality does improve, and generally the performance is very similar for all values of α. For GBAS and MMAS, the best performing ρ_{alg} value is 0.99 for $\alpha > 2$, with 0.9 doing well with lower values of α. However, Ant System prefers 0.9 for all values of α, with 0.99 consistently being the worst performing ρ_{alg} value.

In terms of convergence there is little difference in most of the runs. For GBAS, $\rho_{alg} = 0.9$ is a little later in converging for lower α values. However, as α increases $\rho_{alg} =$

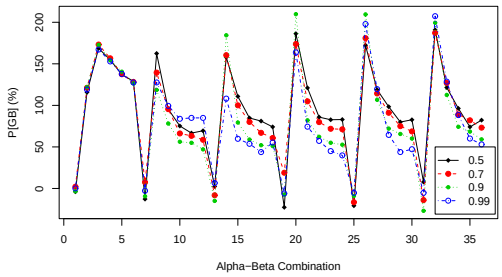
0.99 is the value that delays convergence consistently for approximately 100 iterations, enabling the better performance seen in Figure 6.20(a). For Ant System, there is less variation with the common pattern of $\rho_{alg} = 0.99$ than seen in previous heuristics for this algorithm. $\rho_{alg} = 0.9$ is delaying convergence and this offers an explanation for its better solution quality.

MMAS offers a very different picture. When $\alpha = 1$ and $\beta > 2$ the convergence for low ρ_{alg} is no different from that of NN, but once α moves to 2 these same ρ_{alg} values delay convergence by as much as 400 iterations. As α increases above 2, only $\rho_{alg} = 0.99$ consistently delays convergence for longer than the Nearest Neighbour heuristic, thus the solution quality is better accordingly. A curious phenomena is seen at $\alpha = 3, \beta = 5$ in Figure 6.20(f), at which point there is a small increase in the convergence for all ρ_{alg} values. The reason for this is uncertain although it does not visibly affect the solution quality.

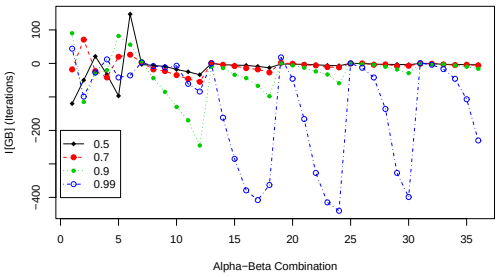
Figure 6.21 gives an example of the Talent Scheduling results. These graphs are very different from those in the previous set, with the biggest difference being that DRI actually gives *better* performance in some cases, such as $\alpha = 3, \beta = 4$ in Figure 6.21(a), which is 600 units better than the Nearest Neighbour heuristic achieved. GBAS and MMAS show a very similar pattern with $\rho_{alg} = 0.5$ achieving the best performance in both. The general performance of each is very complex with a +/- 200 unit difference at any particular stage in the graph. In contrast, for Ant System most of the points are greater than zero indicating that DRI performs worse than NN. Unusually, the peaks indicating the worst performance all occur when $\beta = 2$.

Figure 6.21(b) for GBAS illustrates the variable I_{GB}, showing less difference from the convergence of NN. Again, it is $\rho_{alg} = 0.99$ that is the leading assignment, but it is less pronounced than for the TSP, only 150 iterations compared to 400. Ant System shows similar variety to previous graphs of this type, sometimes better and occasionally worse than the Nearest Neighbour heuristic. As for the TSP, MMAS shows a great leap when $\alpha = 2, \beta > 3$ and $\rho_{alg} < 0.99$, but after that only $\rho_{alg} = 0.99$ shows any difference from NN. This leap in convergence coincides with the respective leap in performance.

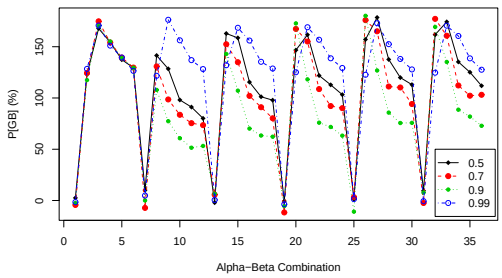
In the significance tests it is expected that the heuristics will show little difference in performance or convergence. There is a case for testing whether DRI performed better than NN, in terms of P_{GB}, for the Talent Scheduling problems, therefore the table will be split into two with these results being tested by a two-tailed test rather than a one-



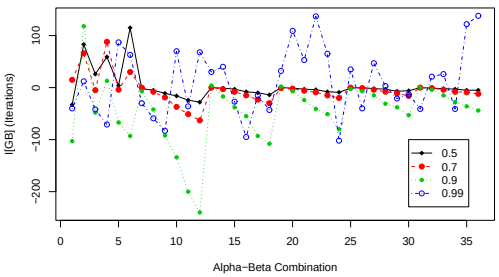
(a) P_{GB} using GBAS



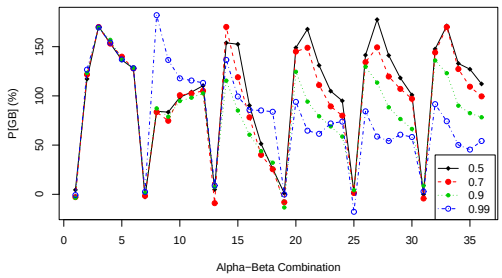
(b) I_{GB} using GBAS



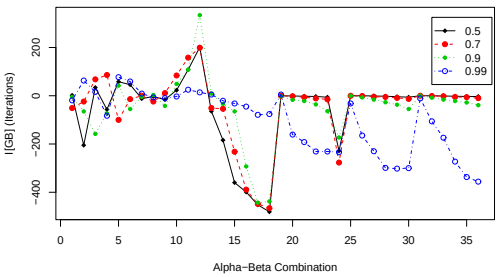
(c) P_{GB} using AS



(d) I_{GB} using AS

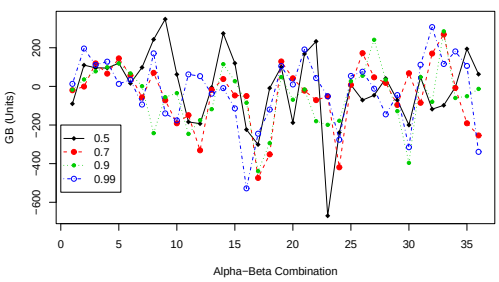


(e) P_{GB} using MMAS

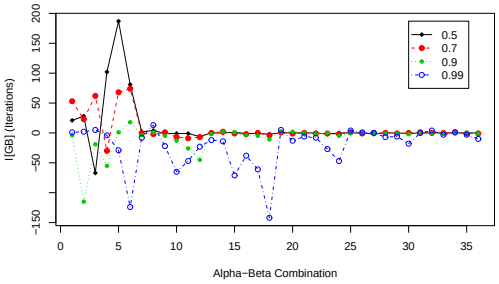


(f) I_{GB} using MMAS

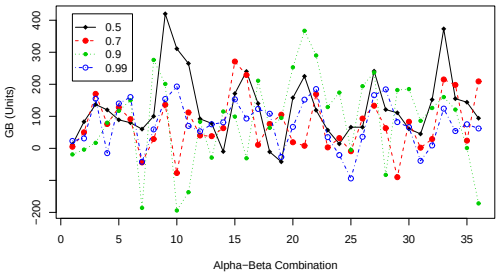
Figure 6.20: Figures illustrating the difference between the DRI heuristic and the NN heuristic for the TSP, problem rd100. (Negative values indicate that DRI performed better for that variable than NN.)



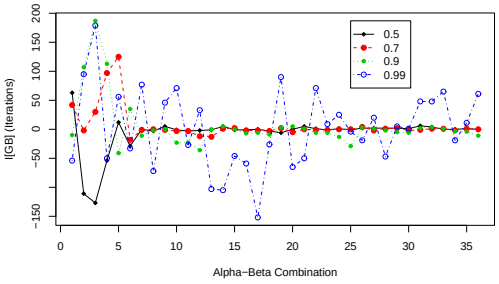
(a) P_{GB} using GBAS



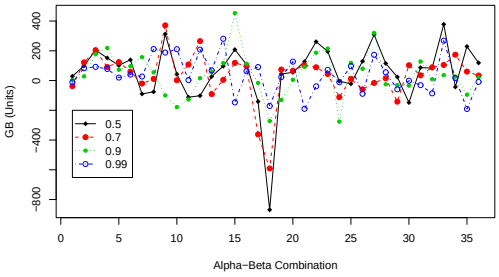
(b) I_{GB} using GBAS



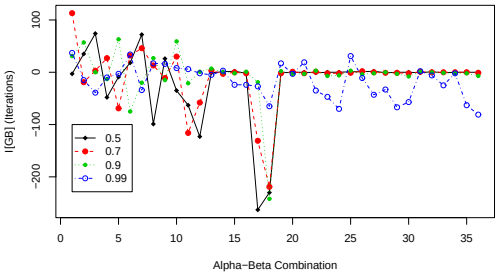
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 6.21: Figures illustrating the difference between the DRI heuristic and the NN heuristic for the Talent Scheduling domain, problem talent_10_100_2. (Negative values indicate that DRI performed better for that variable than NN.)

Problem	ρ_{alg}	P_{GB}			I_GB		
		GBAS	AS	MMAS	GBAS	AS	MMAS
ulysses16	0.5	< 0.0001*	< 0.0001*	0.0512^{*B}	0.0034*	0.0012*	0.0044^{*B}
ulysses16	0.7	0.0002*	< 0.0001*	0.0315^{*B}	0.0034*	0.0016*	0.0033*
ulysses16	0.9	0.0027*	< 0.0001*	0.0315^{*B}	0.0013*	0.0002*	0.0155^{*B}
ulysses16	0.99	0.0283^{*B}	< 0.0001*	0.0009*	0.0005*	0.5627	< 0.0001*
brazil58	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.0019*	0.0012*	0.0063^{*B}
brazil58	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.0018*	0.0009*	0.0032*
brazil58	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.0007*	0.0001*	0.0018*
brazil58	0.99	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*	0.7628	< 0.0001*
rd100	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.0017*	0.0040*	0.0134^{*B}
rd100	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.0053^{*B}	0.0030*	0.0077^{*B}
rd100	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.0049^{*B}	0.0004*	0.0034*
rd100	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.0003*	0.7977	0.0004*
		two-tailed test					
talent_10_20_6	0.5	0.4174	0.4174	0.3135	0.2849	0.4237	0.4819
talent_10_20_6	0.7	0.2303	0.7017	0.3001	0.2292	0.5158	0.2879
talent_10_20_6	0.9	0.1876	0.6810	0.8349	0.1265	0.3634	0.2583
talent_10_20_6	0.99	0.1765	0.3920	0.8174	0.1944	0.7748	0.4087
talent_10_30_12	0.5	0.0297^{*B}	0.1149	0.2648	0.1886	0.2781	0.1661
talent_10_30_12	0.7	0.1401	0.0130^{*B}	0.0912	0.1713	0.1484	0.2862
talent_10_30_12	0.9	0.0946	0.0328^{*B}	0.1188	0.1502	0.2477	0.1913
talent_10_30_12	0.99	0.1149	0.9014	0.1192	0.1264	0.8147	0.2697
talent_10_100_2	0.5	0.8570	0.1839	0.5619	0.3408	0.6824	0.3925
talent_10_100_2	0.7	0.7018	0.3766	0.7956	0.3437	0.4020	0.3175
talent_10_100_2	0.9	0.7160	0.4207	0.6605	0.2862	0.3826	0.2866
talent_10_100_2	0.99	0.9509	0.2649	0.9551	0.2023	0.5715	0.1823

Table 6.6: Table showing the p-values for testing whether the DRI heuristic was worse than the NN heuristic. (A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\frac{0.05}{12} = 0.0042$.)

tailed test. The reason for this change is because although the differences look big they are quoted in units rather than as percentages, which means the differences may appear larger than they actually are.

From table 6.6 one can conclude the following:

- For P_{GB} and GBAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and AS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and MMAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and GBAS on TS, the Alternative Hypothesis can be rejected.
- For P_{GB} and AS on TS, the Alternative Hypothesis can be rejected.
- For P_{GB} and MMAS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and GBAS on TSP, the Null Hypothesis can be rejected.
- For I_{GB} and AS on TSP, the Null Hypothesis can be rejected.
- For I_{GB} and MMAS on TSP, the Null Hypothesis can be rejected.

- For I_{GB} and GBAS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and AS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and MMAS on TS, the Alternative Hypothesis can be rejected.

Therefore given these tests four clear conclusions can be drawn. Firstly, for the TSP, performance is degraded by the DRI heuristic and secondly, that convergence is delayed significantly. This suggests that more search using this heuristic will not result in better quality solutions, although it does not take into account variety of search which might be better than for the Nearest Neighbour heuristic. For the Talent Scheduling problems there is no significant difference between the performance of DRI and NN in either P_{GB} or I_{GB}.

6.3.2.5 FRI compared to NN

Finally, the Fixed Random Increment (FRI) heuristic is compared to the Nearest Neighbour heuristic. Unlike the previous heuristic, FRI adds or subtracts, a fixed amount with equal probability from the original cost returned by the Nearest Neighbour heuristic. FRI is expected to be the most benign heuristic because, although it changes the individual values, the change to the distribution of probabilities over an entire node's outgoing branches is likely to go unnoticed.

In Figure 6.22 the TSP graphs are shown. The graphs of all three algorithms are very similar to those of the previous heuristic, with P_{GB} increasing by about 150% above NN values. For GBAS and MMAS, as α increases so the solution quality decreases. However $\rho_{alg} = 0.99$ performs the best of the ρ_{alg} values, with performance increasing as β tends to 5.

The performance of Ant System is similar, but $\rho_{alg} \in \{0.7, 0.9\}$ are the two values that are best, with 0.9 preferring larger α values. Finally, MMAS shows a different shaped graph to the other two algorithms. When $\alpha = 1$, the performance deviates approximately 40%, from the NN values for $\rho_{alg} < 0.99$. As α increases, the performance degrades to over 100%, except for the ρ_{alg} value of 0.99 which seems to keep at same level thoughout the graph. In all three performance graphs, as β is increased, the solution quality tends to get marginally better.

The graphs for I_{GB} also show very similar things to the previous heuristics. All the convergence results are similar to those of the Nearest Neighbour heuristic, except

for a few observations. The first is the common negative deviation, for GBAS, of $\rho_{alg} = 0.99$, and the unpredictable variation of this same ρ_{alg} value for Ant System. MMAS displays a large increase in I_{GB} for $\alpha = 2, \beta = 2$, although as in the previous heuristic this is not translated into better performance.

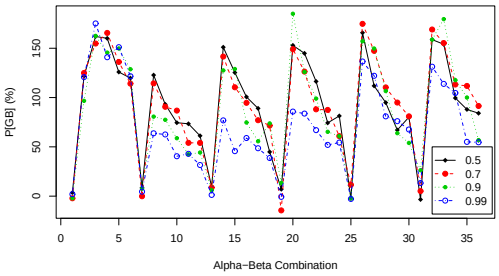
Figure 6.23 illustrates the results for Talent Scheduling. As for the DRI heuristic the P_{GB} graphs show some improvement over the NN heuristic, even when both α and β are high. Having said this, the improvement is not much, only varying by at most 200 units. In the results for the three algorithms all the ρ_{alg} values are similar and vary without an obvious trend.

In terms of I_{GB} the performance again is very similar to NN. The Ant System results show the same variation for $\rho_{alg} = 0.99$ that the other heuristics have shown. The only point of note is, for MMAS when $\rho_{alg} = 0.7$, a large decrease in the convergence of the algorithm is shown at $\alpha = 2, \beta = 3$ for FRI.

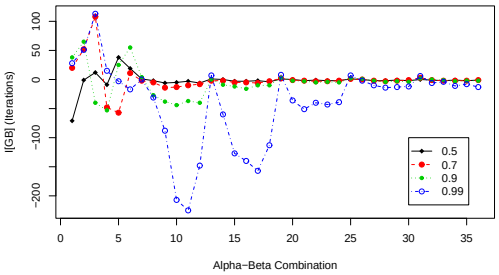
From these graphs it is unlikely any of the significance tests for TS will show any difference. It is expected that the TSP results may show some significant results because the graphs behave similarly to the other heuristics. As was done also with the DRI heuristic for the Talent Scheduling problem, the P_{GB} results will be tested with a two-tailed test instead of the one-tailed, due to the nature of the graphical results.

From Table 6.7 one can conclude the following:

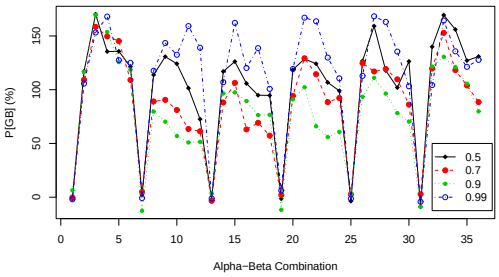
- For P_{GB} and GBAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and AS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and MMAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and GBAS on TS, the Alternative Hypothesis can be rejected.
- For P_{GB} and AS on TS, the Alternative Hypothesis can be rejected.
- For P_{GB} and MMAS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and GBAS on TSP, the Alternative Hypothesis can be rejected.
- For I_{GB} and AS on TSP, the Null Hypothesis can be rejected.
- For I_{GB} and MMAS on TSP, the Alternative Hypothesis can be rejected.
- For I_{GB} and GBAS on TS, the Alternative Hypothesis can be rejected.



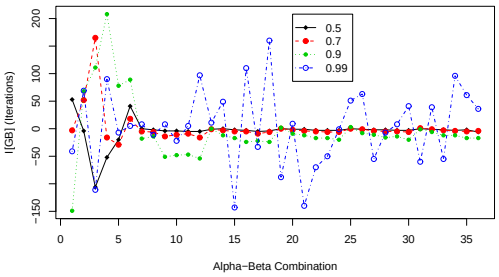
(a) P_{GB}



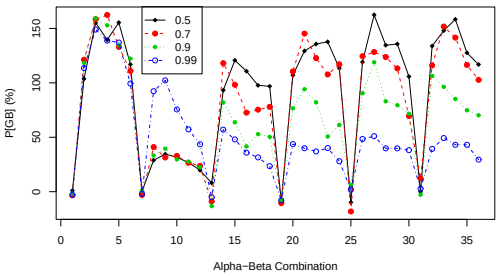
(b) I_{GB}



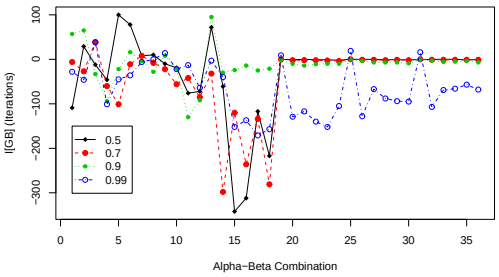
(c) P_{GB}



(d) I_{GB}

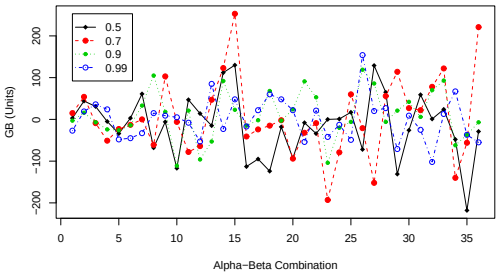


(e) P_{GB}

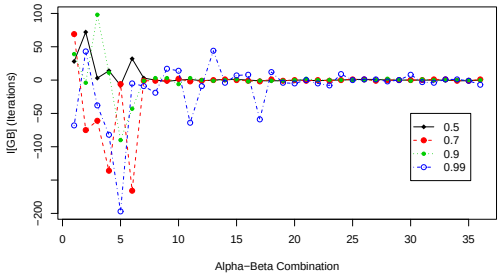


(f) I_{GB}

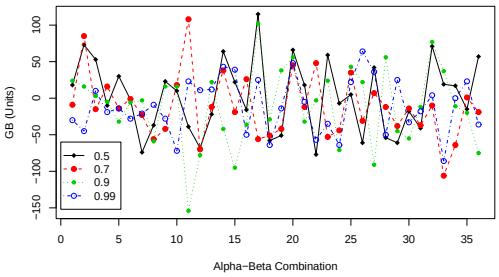
Figure 6.22: Figures illustrating the difference between the FRI heuristic and the NN heuristic for the TSP domain, problem Brazil58. (Negative values indicate that FRI performed better for that variable than NN.)



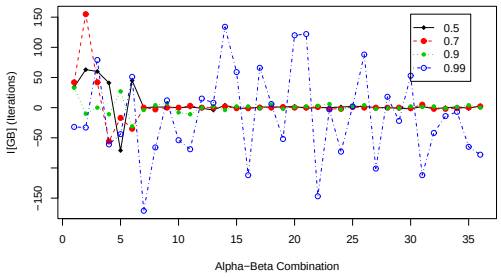
(a) P_{GB}



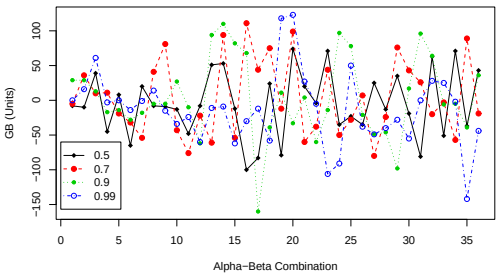
(b) I_{GB}



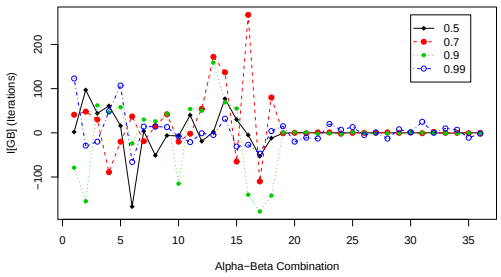
(c) P_{GB}



(d) I_{GB}



(e) P_{GB}



(f) I_{GB}

Figure 6.23: Figures illustrating the difference between the FRI heuristic and the NN heuristic for the Talent Scheduling domain, problem talent_10_30_12. (Negative values indicate that FRI performed better for that variable than NN.)

Problem	ρ_{alg}	P_{GB}			I_GB		
		GBAS	AS	MMAS	GBAS	AS	MMAS
ulysses16	0.5	< 0.0001*	< 0.0001*	0.0039*	0.0498^{*B}	0.0779	0.0837
ulysses16	0.7	< 0.0001*	< 0.0001*	0.0019*	0.0671	0.0799	0.0848
ulysses16	0.9	< 0.0001*	< 0.0001*	0.0007*	0.1033	0.0273^{*B}	0.0633
ulysses16	0.99	0.0001*	< 0.0001*	< 0.0001*	0.0494^{*B}	0.0409^{*B}	0.0196^{*B}
brazil58	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.0366^{*B}	0.0227^{*B}	0.1691
brazil58	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.0397^{*B}	0.0334^{*B}	0.0651
brazil58	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.0836	0.0966	0.0787
brazil58	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.1151	0.6152	0.0064^{*B}
rd100	0.5	< 0.0001*	< 0.0001*	< 0.0001*	0.0156^{*B}	0.0435^{*B}	0.1540
rd100	0.7	< 0.0001*	< 0.0001*	< 0.0001*	0.0514	0.0438^{*B}	0.1011
rd100	0.9	< 0.0001*	< 0.0001*	< 0.0001*	0.0974	0.0445^{*B}	0.1108
rd100	0.99	< 0.0001*	< 0.0001*	< 0.0001*	0.0642	0.8236	0.0285
		two-tailed test					
talent_10_20_6	0.5	0.6893	0.4404	0.9596	0.4603	0.6350	0.8317
talent_10_20_6	0.7	0.8306	0.7228	0.8703	0.5391	0.7967	0.5181
talent_10_20_6	0.9	0.7228	0.6727	0.3412	0.4795	0.5045	0.4596
talent_10_20_6	0.99	0.4539	0.9686	0.9148	0.6174	0.4263	0.5269
talent_10_30_12	0.5	0.9686	0.5965	1.0000	0.5298	0.6953	0.4279
talent_10_30_12	0.7	0.7101	0.7869	0.9461	0.3805	0.4506	0.6437
talent_10_30_12	0.9	0.7101	0.7869	0.9461	0.5317	0.6281	0.4462
talent_10_30_12	0.99	0.9416	0.6768	0.6202	0.3550	0.3002	0.5515
talent_10_100_2	0.5	0.7244	0.4676	0.4495	0.4864	0.5495	0.3557
talent_10_100_2	0.7	0.7697	0.4573	0.6564	0.5158	0.4708	0.4730
talent_10_100_2	0.9	0.9865	0.5064	0.7925	0.4662	0.3282	0.3934
talent_10_100_2	0.99	0.8438	0.3028	0.9282	0.3281	0.2007	0.4462

Table 6.7: Table showing the p-values for testing whether the FRI heuristic was worse than the NN heuristic. (A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\frac{0.05}{12} = 0.0042$.)

- For I_{GB} and AS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and MMAS on TS, the Alternative Hypothesis can be rejected.

Given these tests, four conclusions can be drawn. Firstly, the TSP performance is degraded by the FRI heuristic, and secondly, convergence is not delayed significantly. The convergence results for AS and GBAS were mixed, and although the decision was made with the majority, more evidence would be helpful to support the decisions. For the Talent Scheduling problems there is no significant difference between the performance of FRI and NN in either P_{GB} or I_{GB}.

6.3.2.6 Summary

This section investigated how each heuristic compared to the Nearest Neighbour heuristic in terms of performance and convergence. The results showed a clear difference in the quality of the two Nearest Neighbour heuristics, which explains the fact how two similar heuristics, for two separate domains, can have very different consequences.

For the TSP, the performance of all five heuristics that fed misinformation did worse than the Nearest Neighbour heuristic. This was not an unexpected result as the heuristic used is very popular, and would only become so if it did provide useful guidance. The convergence for the two dynamic heuristics, DRCM and DRI, showed there was a significantly greater number of iterations before the final solution was reached. This demonstrates that using a more active heuristic could be more beneficial in exploration of a search space than a fixed heuristic. All the fixed heuristics (RCM,SRCM and FRI) were found to be no different in their convergence characteristics to the Nearest Neighbour heuristic. This indicates that if one were to design a heuristic to aid exploration it should be reactive and not passive in the algorithm.

The Talent Scheduling problem showed, in contrast to the TSP, how the study of a new application can benefit from misinformation being fed into the algorithm. By comparing using the Nearest Neighbour heuristic to setting $\beta = 0$ and using no heuristic, one would assume that this was the best heuristic possible. However, by using some heuristics that add an element of randomness to the returned values, it is clear that better heuristics might be developed.

For performance DRCM,RCM and SRCM were all significantly worse than the Nearest Neighbour heuristic for the Talent Scheduling problem. Curiously DRI and FRI performed no differently to the Nearest Neighbour, and in fact found *better* solutions than the Nearest Neighbour in some cases. This is probably because the Nearest Neighbour heuristic was in error by a small amount, which the increment fixed. As the range of the costs was only between 1 and 10 this allowed for small increments to make a much larger difference than for the TSP. In terms of significance, none of the heuristics for the Talent Scheduling problems provided different convergence characteristics to the Nearest Neighbour heuristic.

In general, convergence was not affected by the change in heuristic. However, it was affected when $\rho_{alg} = 0.99$, which often increased the number of iterations before the final solution was found. Although this was not unexpected for GBAS, it was surprising how often the same was found for AS and MMAS, given the conclusions of Chapter 5. This can be put down to the choice of ρ_{alg} parameters. If 0.95 had also been chosen it might have shown that 0.99, while performing better than 0.9, would have been worse than 0.95, thus giving the expected trend found in Chapter 5.

6.3.3 Dynamic versus Static Heuristics

Having looked at each heuristic separately and also compared them to the Nearest Neighbour heuristic, a final question remains, "do Ant algorithms handle static heuristics better than dynamic heuristics?" In other words, do Ant algorithms learn when a heuristic is misleading and react to this, which is easier with a fixed heuristic, or do the algorithms prefer changing values where there is always the chance of getting a better heuristic value for a particular move, thereby enabling it to correct itself.

Null Hypothesis:	There is no significant difference between DRCM and SRCM when using the GBAS algorithm for P_{GB} for TSP.
Alternative Hypothesis:	DRCM is significantly greater than than SRCM for the variable P_{GB} when using the GBAS algorithm for TSP.

This hypothesis is repeated for AS and MMAS instead of GBAS, I_{GB} instead of P_{GB} and Talent Scheduling (TS) for TSP. For I_{GB}, the Alternative Hypothesis is a one-tailed test, but it represents better convergence characteristics. For P_{GB}, this hypothesis represents a worse characteristic. This gives a total of twenty-four hypotheses to test, twelve for DRCM compared to SRCM, and twelve for DRI compared to FRI.

Tables 6.8 and 6.9 give the results of the Mann-Whitney U-tests that were conducted. The results are as follows for DRCM compared to SRCM:

- For P_{GB} and GBAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and AS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and MMAS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and GBAS on TS, the Null Hypothesis can be rejected.
- For P_{GB} and AS on TS, the Null Hypothesis can be rejected.
- For P_{GB} and MMAS on TS, the Null Hypothesis can be rejected for small n, but the Alternative Hypothesis should be rejected for large n.
- For I_{GB} and GBAS on TSP, the Null Hypothesis can be rejected.
- For I_{GB} and AS on TSP, the Null Hypothesis can be rejected.
- For I_{GB} and MMAS on TSP, the Null Hypothesis can be rejected.

		P_{GB}		I_GB	
		DRCM < SRCM	*DRI < FRI*	*DRCM > SRCM*	*DRI > FRI*
GBAS					
ulysses16	0.5	< 0.0001*	0.2678	0.0011*	0.0678
ulysses16	0.7	< 0.0001*	0.1116	< 0.0001*	0.0372*^{B}
ulysses16	0.9	< 0.0001*	0.0495*^{B}	0.0003*	0.0141*^{B}
ulysses16	0.99	< 0.0001*	0.0518	< 0.0001*	0.0216*^{B}
brazil58	0.5	< 0.0001*	0.0232*^{B}	0.0002*	0.0296*^{B}
brazil58	0.7	< 0.0001*	0.0128*^{B}	0.0010*	0.0342*^{B}
brazil58	0.9	< 0.0001*	0.0144*^{B}	< 0.0001*	0.0086*^{B}
brazil58	0.99	< 0.0001*	0.0202*^{B}	< 0.0001*	0.0023*
rd100	0.5	< 0.0001*	0.0779	0.0017*	0.0788
rd100	0.7	< 0.0001*	0.0792	0.0023*	0.0736
rd100	0.9	< 0.0001*	0.0696	0.0003*	0.0344*^{B}
rd100	0.99	< 0.0001*	0.0638	< 0.0001*	0.0140*^{B}
talent_10_20_6	0.5	0.0228*^{B}	0.1692	0.0712	0.3333
talent_10_20_6	0.7	0.0006*	0.2236	0.1107	0.2001
talent_10_20_6	0.9	< 0.0001*	0.1163	0.0091*^{B}	0.1380
talent_10_20_6	0.99	< 0.0001*	0.1348	0.0024*	0.1608
talent_10_30_12	0.5	0.0267*^{B}	0.0461*^{B}	0.1167	0.1891
talent_10_30_12	0.7	0.0316*^{B}	0.0716	0.0905	0.2710
talent_10_30_12	0.9	0.0053*^{B}	0.0281*^{B}	0.0302*^{B}	0.1326
talent_10_30_12	0.99	0.0001*	0.0561	0.0034*	0.1853
talent_10_100_2	0.5	0.0844	0.3302	0.2268	0.3815
talent_10_100_2	0.7	0.2478	0.2321	0.1914	0.2974
talent_10_100_2	0.9	0.0395*^{B}	0.3805	0.1345	0.3156
talent_10_100_2	0.99	0.0489*^{B}	0.4440	0.0260*^{B}	0.3488
AS					
ulysses16	0.5	< 0.0001*	0.1497	< 0.0001*	0.0104*^{B}
ulysses16	0.7	< 0.0001*	0.1230	< 0.0001*	0.0079*^{B}
ulysses16	0.9	< 0.0001*	0.0489*^{B}	< 0.0001*	0.0196*^{B}
ulysses16	0.99	< 0.0001*	0.4775	0.6958	0.9482
brazil58	0.5	< 0.0001*	0.0100*^{B}	0.0003*	0.0339*^{B}
brazil58	0.7	< 0.0001*	0.0520	0.0001*	0.0203*^{B}
brazil58	0.9	< 0.0001*	0.0144*^{B}	< 0.0001*	0.0006*
brazil58	0.99	< 0.0001*	0.0131*^{B}	0.5448	0.6386
rd100	0.5	< 0.0001*	0.0258*^{B}	0.0006*	0.0771
rd100	0.7	< 0.0001*	0.0381*^{B}	0.0003*	0.0492*^{B}
rd100	0.9	< 0.0001*	0.0518	< 0.0001*	0.0132*^{B}
rd100	0.99	< 0.0001*	0.0073*^{B}	0.9730	0.3913
talent_10_20_6	0.5	0.0054*^{B}	0.3762	0.0433*^{B}	0.3197
talent_10_20_6	0.7	0.0038*	0.6152	0.0448*^{B}	0.2264
talent_10_20_6	0.9	0.0004*	0.8392	0.0077*^{B}	0.3891
talent_10_20_6	0.99	0.3467	0.9127	0.9993	0.8364
talent_10_30_12	0.5	0.0070*^{B}	0.0260*^{B}	0.0671	0.1903
talent_10_30_12	0.7	0.0053*^{B}	0.0183*^{B}	0.0145*^{B}	0.1617
talent_10_30_12	0.9	0.0030*	0.0324*^{B}	0.0159*^{B}	0.1763
talent_10_30_12	0.99	0.0087*^{B}	0.6195	0.9999	0.9358
talent_10_100_2	0.5	0.2848	0.7075	0.4865	0.5516
talent_10_100_2	0.7	0.5404	0.5269	0.2300	0.4217
talent_10_100_2	0.9	0.3426	0.5715	0.1025	0.5381
talent_10_100_2	0.99	0.1527	0.5649	0.4131	0.7961

Table 6.8: Table showing the p-values for testing whether the dynamic heuristics were better than the static heuristics. (All test results are given as p-values to 4 decimal places. *=Significant at the 0.05 level. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\frac{0.05}{12} = 0.0042$.)

		P_{GB}		I_GB	
		$DRCM < SRCM$	$DRI < FRI$	$DRCM > SRCM$	$DRI > FRI$
MMAS					
ulysses16	0.5	< 0.0001*	< 0.0001*	0.0004*	0.2524
ulysses16	0.7	< 0.0001*	0.1435	0.0005*	0.1020
ulysses16	0.9	< 0.0001*	0.1487	0.0004*	0.1526
ulysses16	0.99	< 0.0001*	0.1066	0.0001*	0.0289^{*B}
brazil58	0.5	< 0.0001*	0.2268	< 0.0001*	0.0479^{*B}
brazil58	0.7	< 0.0001*	0.0353^{*B}	0.0028*	0.0661
brazil58	0.9	< 0.0001*	0.0409^{*B}	0.0003*	0.0134^{*B}
brazil58	0.99	0.0025*	0.0844	< 0.0001*	0.0162^{*B}
rd100	0.5	< 0.0001*	0.3791	0.0004*	0.0823
rd100	0.7	< 0.0001*	0.0353^{*B}	0.0004*	0.0647
rd100	0.9	< 0.0001*	0.0624	0.0005*	0.0234^{*B}
rd100	0.99	0.1297	0.1160	0.0089^{*B}	0.0404^{*B}
talent_10_20_6	0.5	0.0076^{*B}	0.3052	0.0929	0.2359
talent_10_20_6	0.7	0.0178^{*B}	0.2355	0.0090^{*B}	0.2244
talent_10_20_6	0.9	0.0012*	0.1312	0.0081^{*B}	0.2751
talent_10_20_6	0.99	0.0009*	0.4131	0.0018*	0.4374
talent_10_30_12	0.5	0.0518	0.9179	0.0123^{*B}	0.1952
talent_10_30_12	0.7	0.0216*	0.0919	0.0135^{*B}	0.1755
talent_10_30_12	0.9	0.0032*	0.0635	0.0043^{*B}	0.2054
talent_10_30_12	0.99	0.0097^{*B}	0.0581	0.0007*	0.2338
talent_10_100_2	0.5	0.1868	0.0883	0.1768	0.5633
talent_10_100_2	0.7	0.0716	0.4708	0.0412^{*B}	0.3443
talent_10_100_2	0.9	0.0628	0.4798	0.0494^{*B}	0.3978
talent_10_100_2	0.99	0.6759	0.5423	0.0181^{*B}	0.3141

Table 6.9: *continued* Table showing the p-values for testing whether the dynamic heuristics were better than the static heuristics. (All test results are given as p-values to 4 decimal places. *=Significant at the 0.05 level. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\frac{0.05}{12} = 0.0042$.)

- For I_{GB} and GBAS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and AS on TS,the Alternative Hypothesis can be rejected.
- For I_{GB} and MMAS on TS, the Null Hypothesis can be rejected.

From these tests it can be said that, for P_{GB}, all the algorithms show a preference for dynamic heuristics, with their ability to sometimes give the correct cost values for a particular arc. The only complication is for Talent Scheduling and MMAS, when the size of the problem may influence this choice.

For I_{GB} and the TSP the situation is clear, all the algorithms show a difference, demonstrating that the dynamic heuristics allow for greater search by delaying convergence, which is a reason for the better P_{GB} results. For the Talent Scheduling problems, GBAS and AS do not show a significant difference, however MMAS does.

The other comparison is DRI versus FRI and the results are as follows:

- For P_{GB} and GBAS on TSP, the Alternative Hypothesis can be rejected.
- For P_{GB} and AS on TSP, the Null Hypothesis can be rejected.
- For P_{GB} and MMAS on TSP, the Alternative Hypothesis can be rejected.
- For P_{GB} and GBAS on TS, the Alternative Hypothesis can be rejected.
- For P_{GB} and AS on TS, the Alternative Hypothesis can be rejected.
- For P_{GB} and MMAS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and GBAS on TSP, the Null Hypothesis can be rejected.
- For I_{GB} and AS on TSP, the Null Hypothesis can be rejected.
- For I_{GB} and MMAS on TSP, the Alternative Hypothesis can be rejected.
- For I_{GB} and GBAS on TS, the Alternative Hypothesis can be rejected.
- For I_{GB} and AS on TS,the Alternative Hypothesis can be rejected.
- For I_{GB} and MMAS on TS, the Alternative Hypothesis can be rejected.

This comparison shows a different story, which makes any distinction between the two heuristics less clear. For P_{GB}, although there are a few significant cases there is not enough evidence to say that there is a significant difference overall. Therefore for GBAS and MMAS, the Alternative Hypothesis was rejected. For Talent Scheduling

the situation was more distinct, therefore the overall conclusion is that, for P_{GB}, there is no difference in the performance of DRI and FRI. This is not as unexpected as it would appear as an increment to an existing cost would have to be large and varied in its addition/subtraction to the original value to make a big difference to the distribution of heuristic values after normalisation.

For I_{GB}, GBAS and AS show a significant difference between DRI and FRI for the TSP. However, the significant results in I_{GB} do not always correlate with the significant results for P_{GB}, which means later convergence in itself does not necessitate finding a better solution. For MMAS, the difference is not significant for the TSP, but it does seem that high ρ_{alg} values are more likely to give a difference in convergence.

For the Talent Scheduling problem the situation is distinct and there is no difference between DRI and FRI. This is most likely due to the small cost variation in the problems, as opposed to the TSP problems.

6.4 Conclusions

The goal of this chapter was to investigate how misinformation changed the performance of the three Ant algorithms, GBAS, Ant System and the Max-Min Ant System. To do this, extreme heuristics were used to return either completely random information, or to adjust the original cost structure by some random amount. This was done over a number of runs to try and give some idea of how the algorithms would react given such information.

The expected ordering of the heuristics, in terms of performance going from best to worst, was (NN,FRI,DRI,SRCM,DRCM,RCM). The idea being that increments would not affect the distribution of probabilities at any one node, and that as the heuristic information become more random the performance would decline. After studying each heuristic in turn in Section 6.3.1, the expectation changed, so that the dynamic orderings would do better than the fixed random information, resulting in the ordering (NN,DRCM,DRI,SRCM,FRI,RCM). In fact what occurred was that both expected orderings were right to some degree. The first was right in that the incremental heuristics did perform better than the altered cost matrix heuristics. The second ordering was right in respect to the dynamic orderings was better than the fixed orderings. Therefore the correct ordering was (NN,DRI,FRI,DRCM,SRCM,RCM).

Section 6.3.2 showed that sometimes randomness, in addition to a heuristic that is considered beneficial, can result in better performance. In this case it was to improve upon the results of the Nearest Neighbour heuristic in Talent Scheduling, but any new application should do this alongside testing the heuristic against using $\beta = 0$, to see if it can make any difference, and to help in the design of the particular heuristic.

It was expected at the start of this chapter that low ρ_{alg} values results would be beneficial in handling misleading heuristics, as it would allow the pheromone matrix to remove bad information faster and to exploit good information via the reinforcement process. From these results it is possible to conclude that this is in fact not the case, and that high ρ_{alg} values are just as critical when using less accurate heuristics as well as when using good heuristics. This is most likely due to lower ρ_{alg} values making the algorithm converge faster as it tries to exploit the current solution, which at the start is not going to be very good. However, with ρ_{alg} being high this information is kept around, and as a result it allows the algorithm to have time to find better areas to search.

Another important conclusion that has been revealed by these results is that heuristics that are aware of a changing pheromone matrix can be used to not only exploit search areas, but also to explore areas. Rather than having an arbitrary q_0 value it would be more productive to have a heuristic that could feed misinformation to the probability rule of the algorithm to help exploration in a strategic way. This is an interesting topic for further study.

The main conclusion from this chapter is that none of the algorithms handle misinformation in an intelligent manner. If $\beta > 0$ the algorithm is limited to the accuracy of the heuristic in guiding it to good solutions. Therefore, in many instances researchers have just set β to 0, which means that the algorithm cannot use heuristics which may have a lower probability of being right but could be used in certain situations. A better method of incorporating heuristics is to control their influence in some way. A possible method would be to weight them so that they are used to their full potential at the start, while keeping a record of their ability to find better solutions. If this record got worse the weight would go down, if it improved the weighting would be increased.

Equation 6.1 illustrates this idea, where ω_t is the weighting factor for the heuristic. In this equation α and β are removed as they no longer are required. The key is that the weighting applies only to the heuristic, and is time dependent. This method is not the

same as simply changing the equation from a product of the pheromone intensity and the heuristic to a sum of the two.

$$p_{kl}(n,u) = \begin{cases} \frac{[\tau_{kl}(n)]+\omega_t \cdot [\eta_{kl}(u)]}{\sum_{r \notin u,(k,r) \in A}[\tau_{kr}(n)]+\omega_t \cdot [\eta_{kr}(u)]} & l \notin u, (k,l) \in A \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

An alternative method is to not combine them in a direct manner as is currently done. The heuristic is multiplied by the pheromone intensities directly, but one of the problems with this approach is that the heuristic is an undefined function and the pheromone intensities are well-defined. Therefore, a better approach is for the heuristic to suggest the best node to add to the solution, this suggestion is then assigned the same probability as the best node suggested by the pheromone matrix. The distribution of possible nodes to visit is then normalised so that the probabilities add up to 1. The majority of the distribution is then determined by experience of the current problem, but the heuristic is allowed to be judged on the same scale as those nodes in the pheromone matrix. A parameter could then be used to weight the value assigned to the heuristic node in the range $[\tau_{k,worst}, \tau_{k,best}]$, where these are the best and worst values in the pheromone matrix for the choice of next node from node k.

In conclusion, it has been shown that the heuristics are a crucial element in the design of Ant algorithms. It has been shown that Ant algorithms both rely on good heuristics when they are supplied, and that they cannot recover if given a misleading heuristic. This means that in many instances no heuristic is better than using a heuristic that is partially inaccurate. Therefore, heuristics should be given more importance when being used with these algorithms for new applications.

6.5 Summary

This chapter looked at how misinformation could change the performance of Ant algorithms and that the construction and accuracy of the heuristic should be a central design issue. In the next chapter the Ant algorithms are enhanced by combining them with local searches. This continues the topic of how the Ant algorithms react to external knowledge, and also will move the discussion on in Chapter 8 to whether enhancements, such as local search, change not just the performance but also the role of the Ant algorithms in a hybrid search technique.

Chapter 7

An Empirical Investigation of Ant Algorithms with Local Search

Having investigated several Ant algorithms and how they cope with various degrees of misinformation, the thesis continues in this chapter to investigate how algorithms behave when combined with local search. In Chapter 2 nearly all the applications of Ant algorithms cited used some kind of local search method. However, there have not been any rigorous investigations into whether Ant algorithms are really adapted to driving these local search methods. Therefore the goal of this chapter is to provide a detailed study of how the inclusion of local search impacts on the choice of parameters and behaviour of Ant algorithms.

This chapter is a repeat of the experiments from Chapter 5, with the exception that the Ant algorithms are now combined with a local search method. However, many of the arguments that were used in Chapter 5 are still applicable. Therefore, for reasons of clarity these arguments have been repeated.

The chapter begins with an introduction into what local search methods are designed to achieve, followed in Section 7.2 by the methodology specific to this chapter. In Section 7.3 to 7.6 the experiments are described and the results discussed. These are then followed by a discussion of the conclusions made in the chapter.

7.1 Introduction

For many algorithms local search is the main process by which good quality solutions are found. Local search methods provide a deterministic method from moving from an average solution to a local minimum in the search landscape. The role of the metaheuristic that seeds the local search is to find viable solutions for the local search algorithm to act upon. Therefore, the characteristics that make an algorithm useful acting alone may not necessarily be the same characteristics that are good for seeding a local search method.

In this chapter the effect of using local search with the three algorithms GBAS, AS and MMAS is studied. Three algorithms are used, instead of only GBAS, because although Chapter 5 identified common properties between them, it is not necessary that these will hold once the algorithms are combined with local search. In Section 1.2 it was shown that local search methods can easily solve some of the smaller problems in the chosen libraries, therefore a set of larger problems will be used where necessary.

Local search can affect convergence as well as performance, therefore when applied to many Combinatorial Optimisation problems the Ant algorithms are modified to introduce higher diversification. For these tests the *vanilla* versions of the algorithms that were used in Chapter 5 will be used to make comparisons on how behaviour has been modified. The local search methods that will be used are explained in Chapter 3, along with the description of the domain. For the TSP problems the 3-Opt local search will be used because it can be applied to most of the TSP problems with the best results. For the QAP and Talent Scheduling problems their particular 2-Opt algorithms will be used, the latter being described separately in Chapter 4.

To test the algorithms with local search the size of the problems has been taken into account, and the TSP and Talent Scheduling problems have been increased in size. Doing so makes the solution construction process slower, therefore a nearest neighbour parameter (*nn*) is required for the TSP. This parameter is not used for the Talent Scheduling problem as it is not clear that it would be appropriate; for the QAP it is not required as the maximum problem size is 100 locations. The use of this parameter implies that there is an assumption for the TSP which can be stated as follows:

Assumption: For a particular TSP problem it is likely that the optimal solution will contain only arcs that are in the nearest x number of cities.

In this assumption x is an integer less than n but is normally around 8. For these experiments a value of 20 was chosen for two reasons. The first is that the larger value could be applied to a greater number of individual problems. The second reason was that it did not compromise the difficulty of the problem to such a degree that the Ant algorithms were not required. The reference where this assumption is first made is unknown but it is implicit in any paper than includes this nearest neighbour parameter. For this thesis, the assumption is explicitly made for clarification.

The size of the Talent Scheduling problems was controlled in order that the runtime did not become excessive. For the QAP the problems were randomly picked from the same set as in Chapter 5, and were not biased in favour of larger problems because QAP problems are regarded as some of the hardest of the NP-Hard problems.

The rest of the chapter is comprised of an overview of any new experimental conditions that hold for the experiments later on. After this the experiments for each parameter combination will be studied. Following on will be the conclusions from this chapter and a short discussion.

7.2 Experimental Methodology

The following design choices were made for each algorithm.

- The Nearest Neighbour parameter (nn) is equal to n for problems with $n <= 50$ and 20 otherwise. This parameter influences the TSP both in the choices for the Ant algorithms and for the local search.
- TSP problems used the 3-Opt local search, QAP and TS used a 2-Opt local search.
- No parameter, such as q_0, to control exploration versus exploitation.
- $\Delta\tau_{ij} = \frac{1}{1+f(s)}$, as the amount of pheromone to add in accordance with the update rule.
- Only the global best solution(s) will be used to update the pheromone matrix in accordance with the particular algorithm.
- The local search method is only run on the best solution of the iteration.
- max_{it}, the maximum number of iterations the algorithm will run for.

	α	β	ρ	m
α	NA	Yes	Independent	Independent
β	Yes	NA	Independent	Independent
ρ	Independent	Independent	Yes	Yes
m	Independent	Independent	Yes	Yes

Table 7.1: Experimental Parameter Space

- max_t, the maximum amount of time to run each algorithm for.

As in Chapter 5 the algorithms will be compared by varying a fixed number of parameters, which are listed below.

- α, to weight the influence of the pheromone matrix,
- β, to weight the influence of any heuristics used,
- ρ, to regulate as the pheromone decay,
- m, the number of ants per iteration.

All the experiments made the following assumption:

Assumption: The parameters involved in the probability rule are independent of those used in the trail update rule.

This was assumed to cut down the parameter space and via exploratory runs was found to be consistent. In Table 7.1 the space of parameters has been illustrated, showing which are considered independent and which are not.

A second assumption has been made for those experiments involving β being set higher than zero, thereby introducing heuristic information into the algorithm, which is the following:

Assumption: In general, the values returned by the heuristic are valid indicators of good solutions.

This means that the heuristic will be treated like an oracle relaying information that will, in general, lead us to better solutions. Given the results in Chapter 1.2, this is not a large assumption to make, but it does need to be made.

The experiments conducted with the number of ants m were split into two groups. The first a direct setting of m from the range $[1, 2n]$, where n is the number of components

in the problem, and was allowed to run for 500 iterations. The second setting influenced the maximum number of iterations that the algorithm could perform. Equation 7.1 means that the total number of samples taken is 1000, the idea being to test how effective the algorithms were at using a fixed number of samples. This variable will be called m_2.

$$max_{it} = \frac{1000}{m} \tag{7.1}$$

Each experiment is split into two parts one for GBAS and AS, the other for GBAS and MMAS. This eliminates the requirement for more sophisticated statistical reasoning where three or more variables are concerned. Once the correlations are calculated the decision to accept or reject the Alternative Hypothesis agrees with what the majority of the problems result in. No further decision test was done by grouping these correlations in some way as this test would not have any statistical significance. In general if the decision is not distinct there will be further discussion.

Each run in the experiment was repeated 25 times with the following problems sets (n corresponds to the number of cities for the TSP, the number of flows and distances in the QAP, and the number of shooting days in TS):

- 5 TSP which were selected from preliminary experiments to demonstrate effectively how performance changes with n

  kroA150,rat195,gr202,att532,rat575

- 8 QAP problems were selected, two problems were taken at random from each subgroup (see sub-section 3.3.2)

UIGD	nug20, sko100a
URGI	lipa40a,tho150
RLI	esc16g,bur26d
RLLI	tai35b,tai60b

- 5 TS problems, chosen from a new set of problems created for this chapter and for those where $n = 100$ they were chosen from the same library of problems used for Chapter 4. One problem was chosen for each n at random. For these problems the optima are unknown and therefore the P_{GB} will be replaced with the global best (GB).

  talent_10_70_8, talent_10_100_5, talent_10_150_4, talent_10_175_4, talent_10_200_5

The expected results are that the experiments will concur with those of Chapter 5 and that the local search will be deemed not to interfere significantly with how the parameters should be chosen for the Ant algorithm. One can set out a hypothesis for the chapter therefore that:

Null Hypothesis:	Local search does not significantly alter the way the the parameters affect the performance of the algorithms.
Alternative Hypothesis:	Local search does significantly alter the way the parameters affect the performance of the algorithms.

The data variables collected were:

- P_{GB} - percentage distance from global optimum calculated as in Equation 7.2 (also referred to as the performance),
- I_{GB} - index of the last iteration resulting in a solution with a better objective value,
- T_{GB} - time taken to produce the solution closest to the global optimum,

where GB abbreviates Global Best.

$$\frac{\text{global best} - \text{known optimal}}{\text{known optimal}}\% \tag{7.2}$$

The first variable characterises the performance of the algorithm in terms of its ability to reach known optima. Where a known optimum is not known the actual value of the best solution will be used. The advantage of the percentage is that it makes it easier to compare performance over multiple problems. This variable will be referred to as the performance of the algorithm.

The second variable gives an idea of the convergence of the particular algorithm. Convergence itself was not measured for each algorithm as there are multiple definitions as what constitutes convergence for a particular pheromone matrix and as these experiments are trying to deal with the practical implications, this measure was chosen as a valid substitute. Therefore in this chapter when convergence is mentioned this is what is meant.

Finally, time was included for completeness, but no results will be given as the times correlate with I_{GB}.

These variables were compared for correlation by using the Pearson Product-Moment Correlation on the medians of the 25 samples. This correlation was favoured over the Spearman Rank Correlation as there was enough sampling to satisfy the parametric nature of the Pearson Product-Moment Correlation and the data was of a interval or ratio nature. It was also the case that the experiments were designed to make sure they lent themselves to a related paired test satisfying the final condition for the Pearson Product-Moment Correlation test. In the tables laying out the results of these tests, the correlation coefficient (r) is given to two decimal places and the p-value, indicating the significance of the correlation, is given to four.

In each section examples of the graphs produced for each problem will be given. There are obviously more graphs than those shown but the ones chosen illustrate a particular anomaly or were representative of the others. Where the results differed between problem domains figures are included from each. This results in a large number of graphs which for the reason of better presentation are given with any tables at the end of each section.

The figures plot the non-parametric statistics of the results. Therefore the points are set at the median of the samples and the error bars setting out the interquartile range. The reasons for using these non-parametric statistics were given in Chapter 3. Finally, any results that were derived from the original data and broken down to smaller groups will be given as integer percentages. This is because the precision of these figures is no longer significant due to the number of problems chosen for each domain but they indicate certain trends that are then observed in other tables or figures.

7.3 Variation in ρ

In this section the decay rate of the pheromone matrix will be varied to see the effect on the three algorithms. As in Section 5.4 ρ_{alg}, which can take the values $(0,1)$, will be used to represent the proportion of the pheromone matrix from the previous iteration (τ_{t-1}) used in a particular algorithm. In the case of GBAS this is equal to $(1-\rho)$ and in AS and MMAS it is equal to ρ.

The expectation for this experiment was that ρ_{alg} would not be a linear relationship and therefore it was split into the same two groups as for Chapter 5:

- $\rho_{alg}^{low} = (0.0, 0.9]$,

- $\rho_{alg}^{high} = [0.9, 1.0)$.

The hypothesis for this experiment is the following:

Null Hypothesis:	There is no significant correlation between GBAS and AS when varying ρ_{alg} in the range ρ_{alg}^{low} for P_{GB} for TSP.
Alternative Hypothesis:	There is a significant correlation between GBAS and AS when varying ρ_{alg} in the range ρ_{alg}^{low} for P_{GB} for TSP.

The same hypothesis is being tested for MMAS in place of Ant System, for both P_{GB} and I_{GB} and for both ranges of ρ_{alg}, thereby testing 8 hypotheses for each domain. In total 24 hypotheses are tested. The results of testing these hypotheses were expected to agree with the previous experiments so that:

- For $\rho_{alg} \in (0.0, 0.9]$ GBAS and AS are significantly correlated in both their performance and convergence.
- For $\rho_{alg} \in (0, 0.9]$ GBAS and MMAS are significantly correlated in performance but not for convergence.
- For $\rho_{alg} \in [0.9, 1.0)$ both AS and MMAS are not significantly correlated to GBAS.

The parameters of the algorithm were kept similar to those in previous experiments to maximise comparison.

- $\alpha = 1$
- $\beta = 0$
- $m = 10$, in Chapter 5 this was shown to be adequate for good performance by each algorithm.
- For the TSP $nn = 20$ for all problems $n > 50$ and for $n < 50$, $nn = n$.
- $max_{it} = 500$
- $max_t = 300$ or 5 minutes.

In Figure 7.1 examples of the TSP results are illustrated. The first striking observation is that MMAS is best when ρ_{alg} is less than or equal to 0.2. This entails that it is using little or even none of the information from the pheromone matrix. AS and GBAS on the other hand display a curve that fits with expectations, which is that as ρ_{alg} tends to 0.9 the performance improves.

In terms of the convergence of these algorithms, it is clear that MMAS loses its ability to generate interesting solutions for the local search to work on when ρ_{alg} moves above 0.4 and it is from then on highly correlated to AS and GBAS. ρ_{alg}^{high} does show similar behaviour to that expected, which was that GBAS prefers a value of 0.99 as opposed to 0.95, preferred by AS and MMAS.

Sko100a is used as the example for the QAP in Figure 7.2. These four graphs show very similar behaviour to that observed when local search was not used. All three algorithms' performance are similar, and the only point of note is that MMAS again shows a decline in its ability to generate new solutions at $\rho_{alg} = 0.4$ in Figure 7.2(c).

Figure 7.3 shows an example of the results for the Talent Scheduling problems. In contrast to the graphs for MMAS on the TSP problems, for P_{GB} the improvement of MMAS is as expected, and none of the difficulty of producing new solutions exists. Figure 7.3(c) shows the convergence of MMAS above the levels of both AS and GBAS, in a similar way to when no local search was used. For ρ_{alg}^{high}, GBAS is again following the trends of AS and MMAS but at a more gradual decline and its convergence continues to be delayed up to 0.99.

The results for each problem were correlated using the Pearson Product-Moment Correlation and these results are given in Tables 7.3 for P_{GB} and in Table 7.4 for the variable I_{GB}. The former table shows that the performance, when ρ_{alg} is low, for both AS and GBAS is still predominantly correlated. However, focusing on Talent Scheduling reveals only 2 out of the 5 results show a significant correlation. Figure 7.3(a) gives an example of why this difference occurs; although the GBAS and AS results seem tightly coupled, when GBAS strays, Ant System tends to move in the opposite direction.

When ρ_{alg} is increased above 0.9, the TSP problems are no longer significantly correlated. The QAP problems show a mixture of results, with the two largest problems showing significant correlations. Unlike for the lower group, Ant System, when run on the Talent Scheduling problems, is significantly correlated with GBAS.

For MMAS the results are mixed. For neither group of ρ_{alg} are any of the TSP problems correlated. The QAP problems show a more mixed response with one or two problems from each subgroup being significantly correlated. There is no clear trend to these QAP results, with no apparent link with the complexity of the problem in terms of the Flow Dominance property. This is highlighted by the results that {nug20,sko100a} and {tai35b,tai60b} have similar values for the Flow Dominance property, but which

		AS		MMAS	
		ρ_{alg}^{low}	ρ_{alg}^{high}	ρ_{alg}^{low}	ρ_{alg}^{high}
TSP	P_{GB}	N	A	A	A
	I_{GB}	N	A	A	N
QAP	P_{GB}	N	A	A	A
	I_{GB}	N	A	A	A
TS	P_{GB}	A	N	A	N
	I_{GB}	N	A	A	A

Table 7.2: Table showing which ρ_{alg} hypotheses have been rejected. (N=Null Hypothesis rejected, A=Alternative Hypothesis rejected)

are polar in terms of significance. The Talent scheduling problems show a low response to ρ_{alg}^{low} but are all significantly correlated as ρ_{alg} increases above 0.9.

For the variable I_{GB} in Table 7.4, the picture is more distinct. It can be observed that for low ρ_{alg} values AS and GBAS are significantly correlated, but for ρ_{alg}^{high} the correlations are no longer significant. For MMAS, neither group of values shows much significant correlation except for a few TSP problems and a single QAP problem.

These correlations lead to the conclusions shown in Table 7.2.

When comparing these conclusions to those expected there are several points to discuss. The primary difference is the complexity of these conclusions with those of Section 5.4. In the results gained without local search the conclusions could be made per grouping for all the domains because the results were not mixed. With the introduction of local search the conclusions for each algorithm are much more specific to each domain and ρ_{alg} group.

The conclusions for the Ant System algorithm do agree in most respects with the conclusions reached when no local search method was used. Performance and convergence with local search is correlated for ρ_{alg}^{low}, with the exception of Talent Scheduling. For ρ_{alg}^{high}, Ant System was not significantly correlated to GBAS when no local search was used, and again the results in Table 7.2 agree with this, apart from Talent Scheduling.

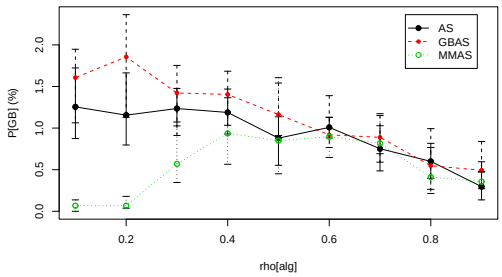
The conclusions for MMAS and GBAS differ to those without local search. For the TSP the conclusion is that the algorithms are not significantly correlated for either performance or convergence for ρ_{alg}^{low}. However, for ρ_{alg}^{high} the problems are significantly

Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
$\rho \in (0, 0.9]$			
TSP	5	5	0
QAP	8	8	4
TS	5	2	2
$[0.9, 1)$			
TSP	5	0	0
QAP	8	3	4
TS	5	5	5

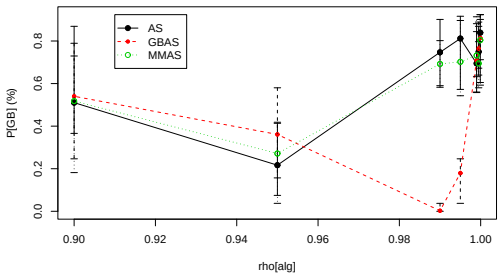
Table 7.3: Table summarising the hypothesis test results for P_{GB} varying ρ_{alg}. (For more detailed results refer to Table C.1.)

correlated in terms of their convergence trends. The performance of the algorithms on the QAP problems, for both ρ_{alg}^{low} and ρ_{alg}^{high}, was evenly mixed, thus the safer conclusion was assumed to hold. This means that the conclusion for the performance does not agree with the conclusion reached when no local search was used. For the variable I_{GB} there were significant correlations between MMAS and GBAS for the TSP when ρ_{alg} was set in the range ρ_{alg}^{high}, however for the rest of the results there was no significant correlation.

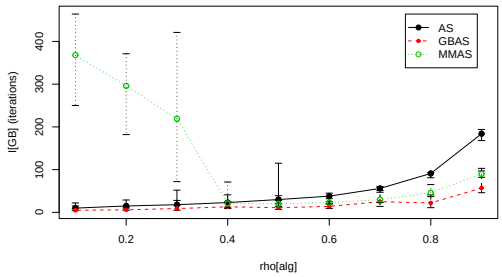
Although many of the conclusions drawn are the same, it appears that the correlations get weaker and become more domain dependent. In the case of Talent Scheduling for instance, both AS and MMAS are now correlated significantly with GBAS in their performance for high values of ρ_{alg}. It is still evident however that a division still exists between the high values of $\rho_{alg} >= 0.9$ and those less than this. The effect of the local search methods is to specialise the parameter settings for each problem, the consequence of which means that optimal parameter settings are much harder to find and to justify.



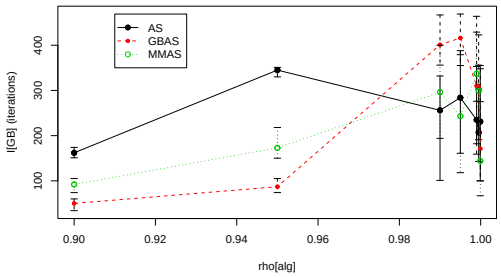
(a) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$



(b) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$



(c) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$

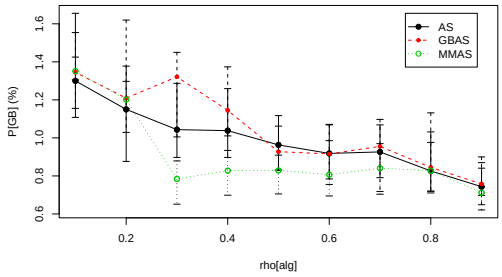


(d) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$

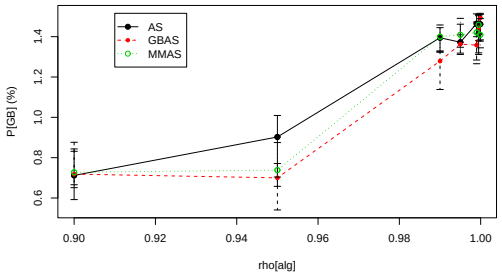
Figure 7.1: Example graphs of the performance of GBAS, AS and MMAS on a TSP problem, in this case gr202, while varying ρ_{alg}. Figures (a) and (c) on the left-hand side are in the range $(0, 0.9]$ while (b) and (d) are in the range $[0.9, 1.0)$.

Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
$\rho \in (0, 0.9]$			
TSP	5	5	0
QAP	8	7	0
TS	5	5	0
$[0.9, 1)$			
TSP	5	0	3
QAP	8	0	1
TS	5	0	0

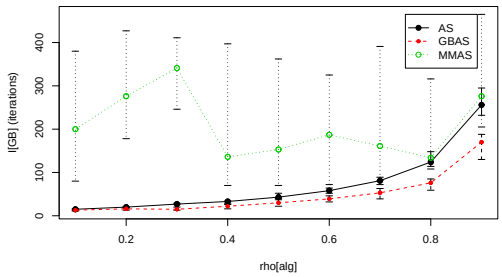
Table 7.4: Table summarising the hypothesis test results for I_{GB} varying ρ_{alg}. (For more detailed results refer to Table C.2.)



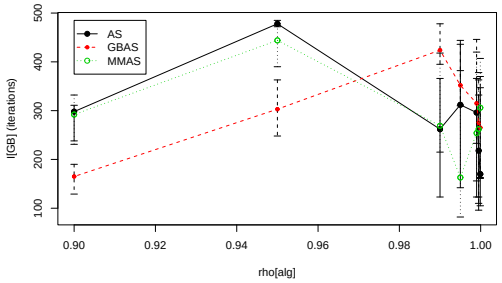
(a) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$



(b) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$

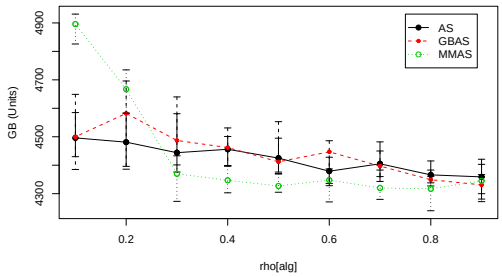


(c) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$

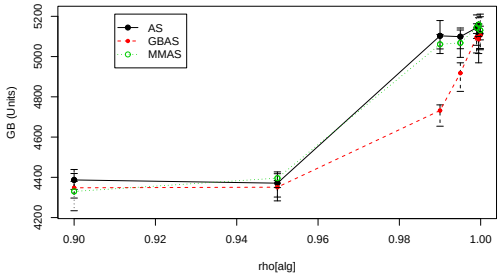


(d) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$

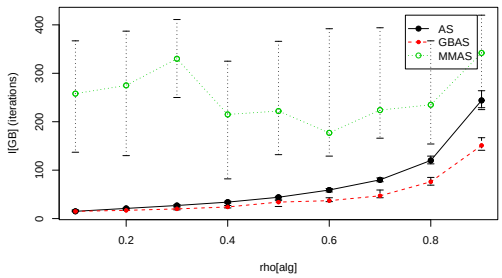
Figure 7.2: Example graphs of the performance of GBAS, AS and MMAS on a QAP problem, in this case sko100a, while varying ρ_{alg}. Figures (a) and (c) on the left-hand side are in the range $(0, 0.9]$ while (b) and (d) are in the range $[0.9, 1.0)$.



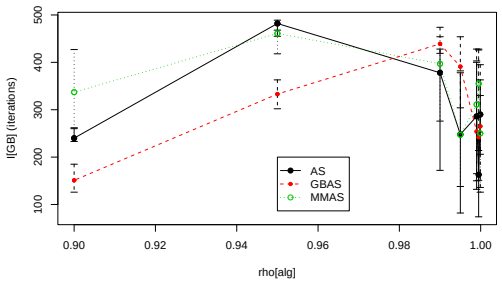
(a) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$



(b) P_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$



(c) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{low}$



(d) I_{GB} varying $\rho_{alg} \in \rho_{alg}^{high}$

Figure 7.3: Example graphs of the performance of GBAS, AS and MMAS on a TS problem, in this case talent_10_100_5, while varying ρ_{alg}. Figures (a) and (c) on the left-hand side are in the range $(0, 0.9]$ while (b) and (d) are in the range $[0.9, 1.0)$.

		AS		MMAS	
		α	β	α	β
TSP	P_{GB}	A	N	N	N
	I_{GB}	A	N	A	N
QAP	P_{GB}	A	–	A	–
	I_{GB}	A	–	A	–
TS	P_{GB}	A	N	A	N
	I_{GB}	A	A	A	A

Table 7.5: Table from Section 5.5 for parameters α and β showing which hypotheses were rejected. The algorithms in these experiments were used without any local search methods. (N=Null Hypothesis rejected, A=Alternative Hypothesis rejected)

7.4 Variation in α and β

The parameters α and β are used to control how the relevant information sources are weighted together when deciding which node to visit next in the construction graph. α controls the weighting of the pheromone intensity (τ), while β controls the weighting of the heuristic (η). When both these are zero, the search is the standard random search, as each is then increased with respect to the other the validity of the values used takes on greater significance.

The objective of this experiment is to see how manipulating these two parameters alters the performance of the particular algorithms, and to see if there are correlations between GBAS and the two other implementations. There are two trends that will expected to be observed which are as follows:

- As $\alpha \rightarrow 5$ performance will at first increase, but then decrease as the number of iterations for the algorithm to converge is reduced.
- As $\beta \rightarrow 5$ performance will increase, but at the cost of faster convergence, which for more complicated landscapes may be a problem.

It is expected that the three algorithms will continue to be correlated as shown in Table 7.5 when no local method was used.

There are twenty hypotheses that are being tested in this chapter and are outlined in the one below. Variations include testing I_{GB} instead of P_{GB}, by domain and by α and β.

Null Hypothesis:	There is no significant correlation between GBAS and AS when varying α and β in the range $[0,5]$ in P_{GB} for TSP.
Alternative Hypothesis:	There is a significant correlation between GBAS and AS when varying α and β in the range $[0,5]$ in P_{GB} for TSP.

To test how the three algorithms perform with relation to these two parameters, the following conditions were set:

- $m = 10$,
- $\rho_{alg} = 0.95$
- $max_{it} = 500$.

ρ_{alg} was set at 0.95 as this is the value a large number of the previous applications used. Therefore it was chosen make the experiment applicable to previous work, even though it is in the area where the algorithms do not show significant correlation. The value of ρ_{alg} does not affect the comparison anyway as it is assumed to be independent of α and β.

α and β were then varied in the range $[0,5]$, creating 36 tests per problem and with 25 runs per test set, this creates 9,000 (36 values combinations * 25 samples * 10 problems) individual experiments comprising the TSP and Talent Scheduling problems. For the QAP problem set, only α was varied as no heuristic is used for these, resulting in 1,200 experiments (6 values * 25 samples * 8 problems). In total this makes 10,200 tests.

Figure 7.7 shows an example of how the performance of the algorithms varied with different α values. These results show there is little difference in the results between the various β values and that the local search has broken the link between the heuristic output. Meanwhile the convergence of the algorithms has remained similar to when no local search had been used.

In Figure 7.8, which shows the Talent Scheduling results while varying α, there seems to be slightly more difference between the various β settings, but this may be due to the fact that the global best is used rather than percentage from the known optimum. The convergence patterns are also very different for this problem. As α and β increase, so I_{GB} seems to increase, meaning that the algorithm takes longer to stagnate prematurely at a local optimum with high parameters. This is in contrast to when no local search

was used, where I_{GB} dropped rapidly as the parameters were increased.

Figure 7.9 illustrates the results for the QAP. These graphs show similar trends to the Talent Scheduling problems when $\beta = 0$. For these problems, MMAS has the ability to perform equally well with various α values. It is clear, for all three algorithms, that using no heuristic is made a more viable option when using a local search method because, in general, the heuristic is less important in the performance of the algorithm.

The independence that the local search method creates between value of β and the performance of the algorithm is shown clearly for the TSP in Figure 7.10. The almost linear relationship between all the α curves show how dramatic this independence is. The results agree with those in Section 5.5, which did not use local search, that the best setting for α was 1. For Ant System in Figure 7.10(d), as α is increased the convergence also increases, in contrast to both GBAS and MMAS where the opposite is true.

Figure 7.11 represents the Talent Scheduling problem results. In all three algorithms, when $\alpha \in \{1,2\}$, the trends seem to be affected by β, however out-with this set the trends return to a linear pattern. The graphs for the variable I_{GB} are much more entangled than their TSP counterparts. For all three algorithms, the convergence is delayed for greater values of β, except for $\alpha = 1$ for MMAS which has a pattern all of its own. This situation is very different from the graphs in Section 5.5, which were much more predictable and distinct. This is evidence of the local search making the behaviour of the Ant algorithms more unpredictable.

As TSP employs a 3-Opt local search and the others employ a 2-Opt search, these results indicate that the stronger the local search is, the greater the independence between β and the quality of the final solution. To confirm this result a TSP problem was chosen that had shown this independent behaviour and run with both 2-opt and 3-opt to see the variation in performance as β was varied. The problem chosen was a TSP problem, gr202. It was chosen as it was a large problem capable of being solved by both 2-opt and 3-opt algorithms and also showed the behaviour being studied.

The results are shown in Figure 7.4. The same independence between the varying β values exists for both 2-Opt and 3-Opt. In addition there appears to be a diminishing of the variability of the lines as one moves from 2-Opt to 3-Opt. For instance, the line $\alpha = 3$ in all algorithms for the 2-Opt algorithm has an increase in P_{GB} as β tends to 5, but in the same graphs using 3-Opt these lines have an increased number of inflections

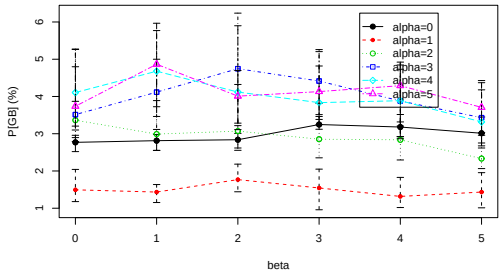
which brings the line back across the midpoint. This is an indication of the increasing independence of β.

To solidify this claim an experiment was then performed using the worst heuristic of Chapter 6, which was the *Reverse Cost Matrix* (RCM). The TSP and TS problems were re-run using this heuristic and then compared with those from using the Nearest Neighbour (NN) heuristic for each problem type. Without local search this experiment would have expected to display a degradation in performance of up to 500%. It should be noted that the local search method does use the real cost matrix for its calculations; the idea being that one has a reliable local search method, but being driven by an Ant algorithm using an unreliable heuristic. The 25 runs of each problem per heuristic were then compared using the Mann-Whitney U-test using paired values, which were the medians of the distributions for of each individual point. The hypotheses used were as follows:

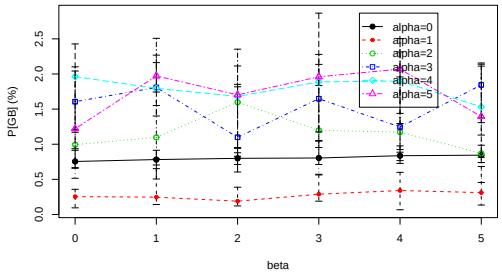
Null Hypothesis:	There is no significant difference between using the Reverse Cost Matrix heuristic and the Nearest Neighbour heuristic when varying α and β in the range $[0,5]$ with local search.
Alternative Hypothesis:	There is a significant difference between using the Reverse Cost Matrix heuristic and the Nearest Neighbour heuristic when varying α and β in the range $[0,5]$ with local search.

Table 7.6 shows that for GBAS all but one of the problems is not significantly different. AS and MMAS show a greater dependence on β, demonstrated by the increase in the number of problems that reject the Null Hypothesis. However in Figure 7.5, not only is the difference in performance shown to be only tenths of a percent compared to 500% without the local search, but for many of the (α,β) combinations, RCM does *better* than the Nearest Neighbour heuristic. This shows how much the local search method contributes to the performance of the algorithm as a whole.

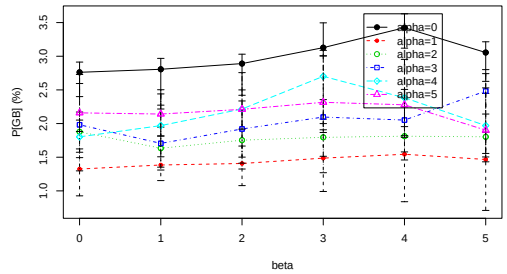
Figure 7.6 shows the difference between the heuristics for the Talent Scheduling problem. These results, unlike those for the TSP, are less in favour of RCM, with combinations such as $(\alpha,\beta) = (1,2)$ to $(1,5)$ for the GBAS algorithm being as bad as when no local search was used. However, as α increases both the performance and convergence tend to fall to about zero. In one extreme case, for Ant System the performance



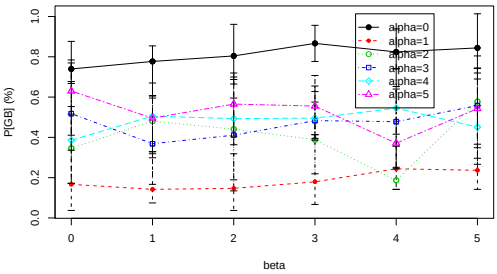
(a) P_{GB} using GBAS (2-Opt)



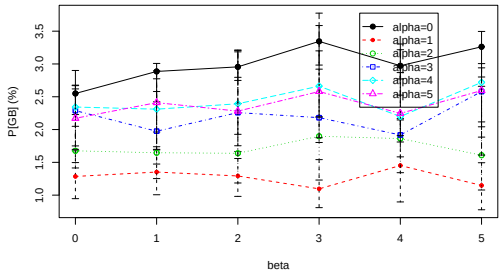
(b) P_{GB} using GBAS (3-Opt)



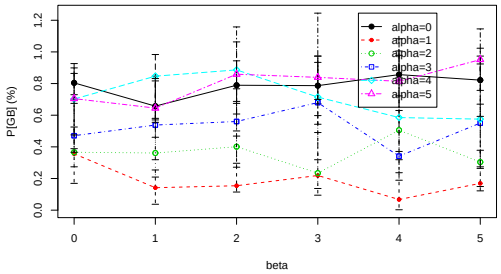
(c) P_{GB} using AS (2-Opt)



(d) P_{GB} using AS (3-Opt)



(e) P_{GB} using MMAS (2-Opt)



(f) P_{GB} using MMAS (3-Opt)

Figure 7.4: A comparison of results for a TSP problem (gr202) using 2-Opt and 3-Opt local search varying β over GBAS, AS and MMAS.

Problem	GBAS	AS	MMAS
kroA150	0.3426	0.9687	0.2119
rat195	0.5390	0.0182^{*B}	0.8985
gr202	< 0.0001*	0.0120^{*B}	0.0005*
att532	0.2088	0.0733	0.5349
rat575	0.1477	0.0041*	0.0061^{*B}
talent_10_70_8	0.5143	0.0011*	0.0003*
talent_10_100_5	0.4609	0.0267^{*B}	0.0218^{*B}
talent_10_150_4	0.4555	0.0009*	< 0.0001*
talent_10_175_4	0.9187	0.0020*	0.0149^{*B}
talent_10_200_5	0.5349	0.1433	0.0081^{*B}

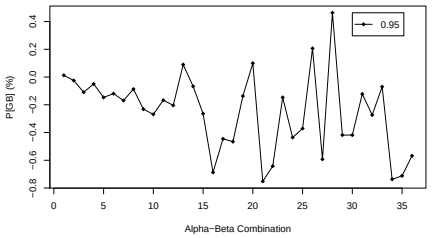
Table 7.6: Table showing the p-values for the significant difference of P_{GB} between using the Reverse Cost Matrix heuristic and the Nearest Neighbour heuristic. (*=Significant difference, p-value < 0.05. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs is $\alpha = \frac{0.05}{10} = 0.0050$.)

degrades by almost 1200 units for $(\alpha, \beta) = (3, 5)$ but this is very rare, and also when put in context of the scale of the objective values may be a small percentage change.

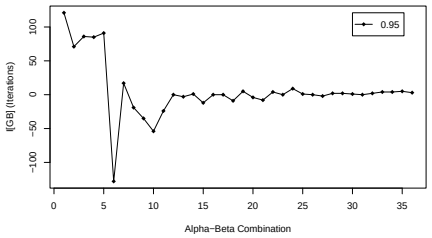
In Table 7.7 the results for the variable I_{GB} give similar conclusions. Overall the conclusion is that GBAS does show the greatest amount of heuristic independence, and that Ant System and MMAS are more susceptible to the heuristic. Figures 7.5 and 7.6 show that for the TSP there is a difference but sometimes that difference can be for the better, and that for the Talent Scheduling problem the choice of heuristic makes little practical difference to the convergence of the algorithm. The figures also illustrate that the differences between the uses of the two heuristics are small even in those that show significant differences in relation to the statistical test.

Where there is better performance, this can be explained by the fact that using the RCM heuristic makes the Ant algorithm search for solutions that are not around the current optimum, which in turn gives the local search something new to optimise. Therefore the local search will find the same local minimum less often as a result, and so the bad heuristic makes the local search *more* productive.

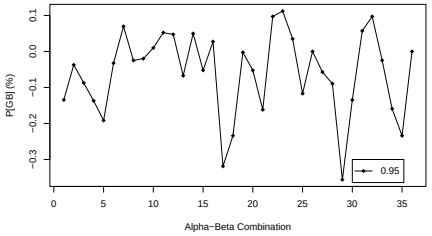
Having looked at this phenomenon, Tables 7.9 and 7.10 show how the rankings of the points of AS and MMAS compare to those of GBAS. High values in these tables indicate that manipulating the parameters will have a similar effect for each algorithm with respect to the other values of that parameter. The percentages in Table 7.9 vary between the three domains. For the TSP, MMAS seems to have the more points in common with GBAS, although these values are not very large. In contrast, the QAP shows that both algorithms behave differently with respect to their other parameter values. The best of



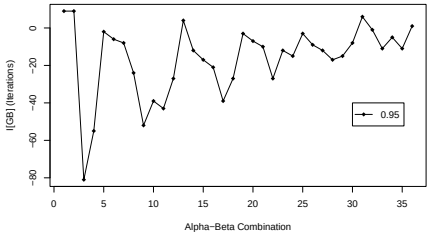
(a) P_{GB} using GBAS (gr202)



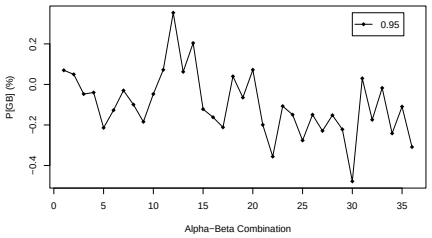
(b) I_{GB} using GBAS (att532)



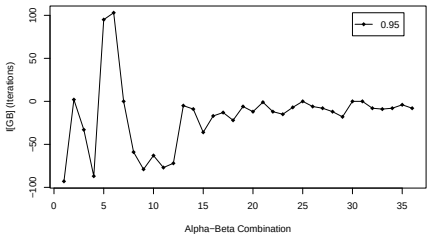
(c) P_{GB} using AS (gr202)



(d) I_{GB} using AS (att532)



(e) P_{GB} using MMAS (gr202)

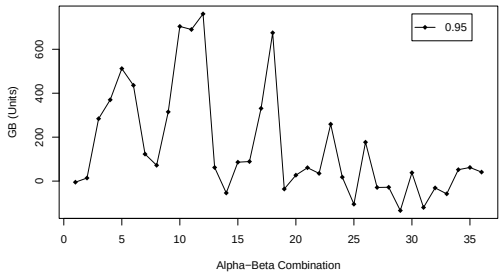


(f) I_{GB} using MMAS (att532)

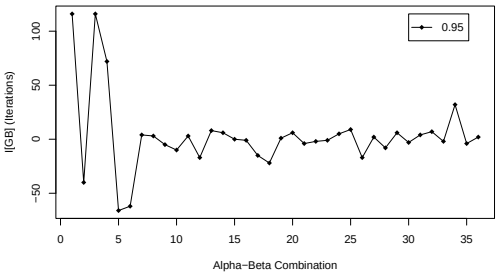
Figure 7.5: Examples of results achieved for taking the difference between the Reverse Cost Matrix heuristic and the Nearest Neighbour heuristic for the TSP. Left: P_{GB} results for the gr202 problem. Right: I_{GB} results for the att532 problem. X-axis shows (α,β) from (0,0) to (5,5), 36 combinations increasing β first. Negative values are in favour of RCM.

Problem	GBAS	AS	MMAS
kroA150	0.0554	0.0227^{*B}	< 0.0001*
rat195	0.0105^{*B}	0.0050*	0.0020*
gr202	0.0001*	0.0004^{*B}	0.0003*
att532	0.4840	< 0.0001*	0.0002*
rat575	0.2637	0.0109^{*B}	< 0.0001*
talent_10_70_8	0.8137	0.0410^{*B}	0.2930
talent_10_100_5	0.4366	0.0020*	0.7119
talent_10_150_4	0.3878	0.2284	0.1818
talent_10_175_4	0.9869	0.9282	0.0282^{*B}
talent_10_200_5	0.4273	0.1483	0.0798

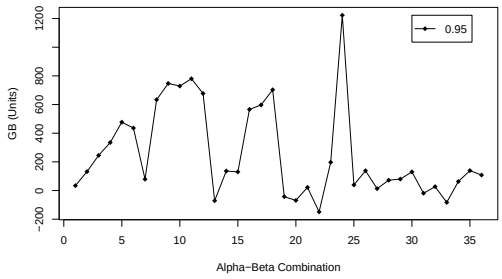
Table 7.7: Table showing the p-values for the significant difference of I_{GB} between using the Reverse Cost Matrix heuristic and the Nearest Neighbour heuristic. (*=Significant difference, p-value < 0.05. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs is $\alpha = \frac{0.05}{10} = 0.0050$.)



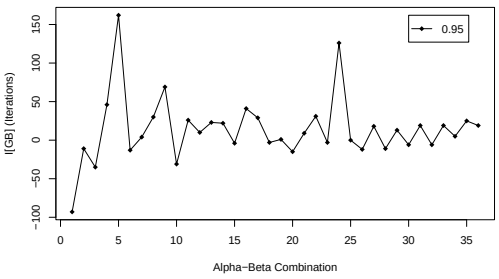
(a) P_{GB} using AS (talent_10_150_4)



(b) I_{GB} using AS (talent_10_175_4)



(c) P_{GB} using MMAS (talent_10_150_4)



(d) I_{GB} using MMAS (talent_10_175_4)

Figure 7.6: Examples of results achieved for taking the difference between the Reverse Cost Matrix heuristic and the Nearest Neighbour heuristic for Talent Scheduling. To the left P_{GB} results for the talent_10_150_4 problem and to right I_{GB} results for the talent_10_175_4 problem. X-axis shows (α,β) from (0,0) to (5,5), 36 combinations increasing β first. Negative values are in favour of RCM.

the three domains is Talent Scheduling that has a number of values greater than 50%, which shows that AS and MMAS have some similarities to GBAS when it comes to the relative settings of parameters. The low figures for P_{GB} in Table 7.10, for both AS and MMAS, indicates a difference in their behaviours when compared to GBAS.

These results compare unfavourably to those gathered when local search was not used, which showed higher similarity in rankings for all domains in Table 5.7. This would seem to indicate that the local search methods alter the behaviour of the algorithms such that a change in these parameters no longer can be predicted across the algorithms.

The Pearson Product-Moment Correlation results are very different from those in Chapter 5 when no local search was used. In Table 7.11 the correlations for varying α while recording the variable P_{GB} is shown. This table shows that only Talent Scheduling demonstrates any correlation between the AS and GBAS. The single QAP result is because the problem was very simple and the local search solved it every time for all the algorithms. For β, shown in the lower half of Table 7.11, the picture is different again. This time there are a few sporadic correlations, but no problem set or particular group stands out. Going back to the graphs reveals that although many of the curves are almost straight, when GBAS performance decays slightly, AS will improve slightly, and so the correlation relationship is weakened.

Table 7.11, shows that when α is varied for MMAS similar results for P_{GB} to Ant System are produced. Large number of significant correlations in the Talent Scheduling set, predominantly as n increases above 70, but this time there are a few TSP correlations in kroA150, which is the smallest problem for the TSP. In the lower far-right column of Table 7.11 shows the results when β is varied and a number of sporadic correlations are found. Therefore whether a correlation is significant or not, appears to be dependent more on the algorithm and problem when a local search method is present than without.

In Table 7.12 when comparing AS and GBAS, in terms of I_{GB}, there are sporadic correlations in TSP and Talent Scheduling with no obvious trends. On the other hand, MMAS is highly correlated for large numbers of problems in all three problem sets, especially and unusually QAP and TSP. Only those runs with $\beta < 3$ are correlated for Talent Scheduling because convergence is increased for MMAS and lowered for GBAS as β moves closer to five. Finally, the table shows that higher α values for the Talent Scheduling are more correlated with TSP and QAP showing no significance

		AS		MMAS	
		α	β	α	β
TSP	P_{GB}	A	A	A	A
	I_{GB}	A	A	N	A
QAP	P_{GB}	A	–	A	–
	I_{GB}	A	–	N	–
TS	P_{GB}	N	A	N	A
	I_{GB}	A	N	A	N

Table 7.8: Table of which hypotheses has been rejected for Section 7.4. (N=Null Hypothesis rejected, A=Alternative Hypothesis rejected)

once again.

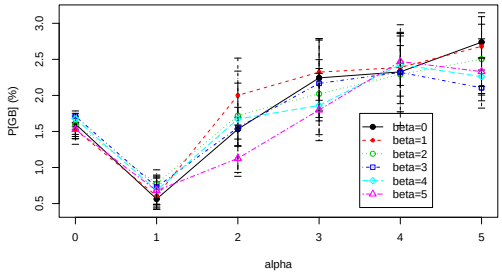
From these significance tables Table 7.8 illustrates the conclusions that can be drawn, with reference to the hypotheses outlined at the start of the section.

There are two conclusions that could be drawn but are not significant due to the quantity of evidence. The first is that by varying α for the Talent Scheduling problem the Max-Min Ant System can be considered to be correlated to GBAS for $\beta < 3$ for P_{GB}. The second is that when varying β for I_{GB}, again for the Talent Scheduling problem, the correlations are significant for $\alpha > 2$ when comparing MMAS and GBAS.

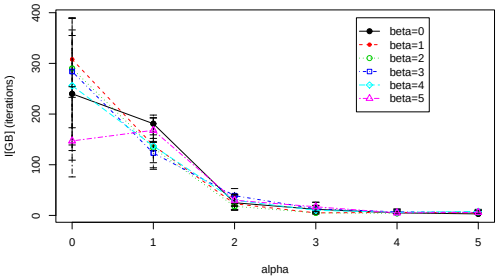
Given these hypothesis rejections and comparing it to Table 7.5 at the start of the section, one can identify 13 out of the 20 hypotheses that have changed. From this one can conclude that local search does have an impact on the parameter settings of α and β. The primary difference is that β no longer results in significant correlations in either Ant System or MMAS. The ability to predict the relationship when varying α has also changed, but remains uncorrelated.

The three conclusions from this section are as follows:

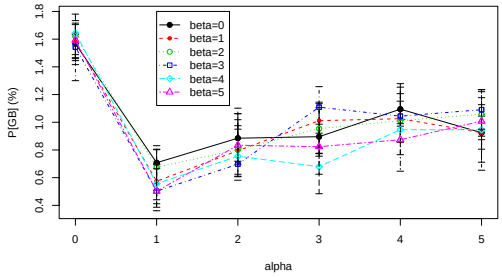
- Local search methods do affect the behaviour of the parameters α and β.
- The strength of the local search method is critical to the impact of the method on the behaviour of the Ant algorithm.
- Local search methods make the ability of the algorithms to perform as expected more dependent on the particular problem being solved.



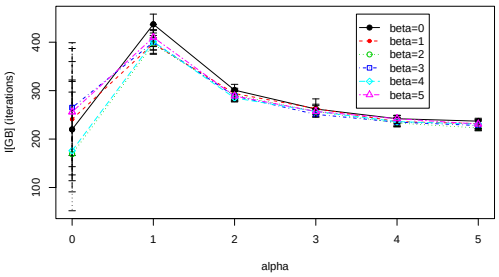
(a) P_{GB} using GBAS



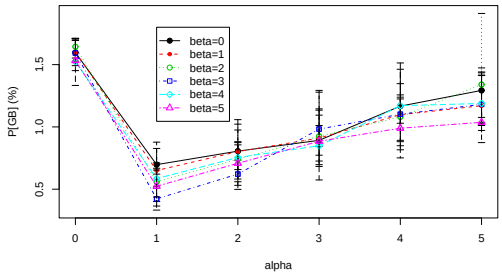
(b) I_{GB} using GBAS



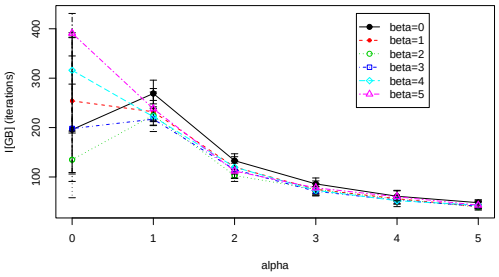
(c) P_{GB} using AS



(d) I_{GB} using AS

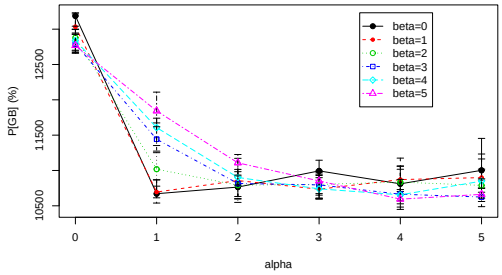


(e) P_{GB} using MMAS

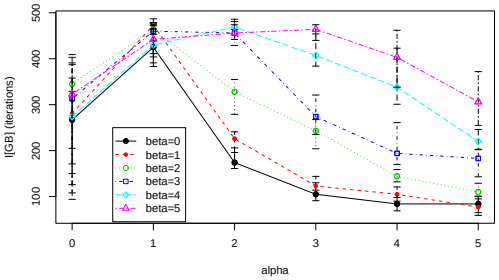


(f) I_{GB} using MMAS

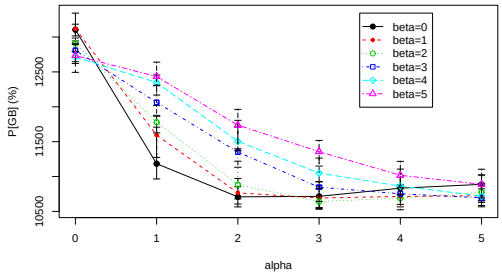
Figure 7.7: Example of the results achieved by varying α over GBAS, AS and MMAS on the TSP data set using att532.



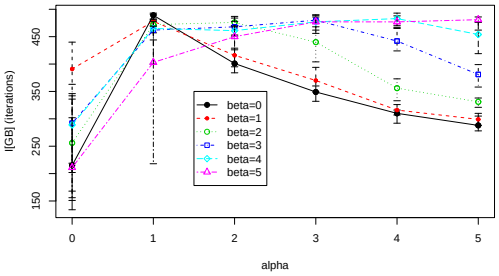
(a) P_{GB} using GBAS



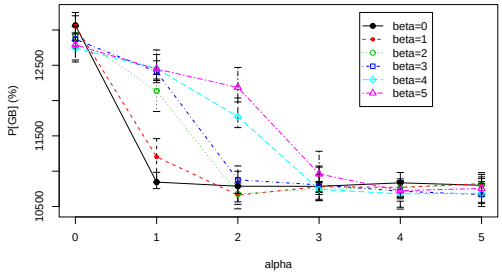
(b) I_{GB} using GBAS



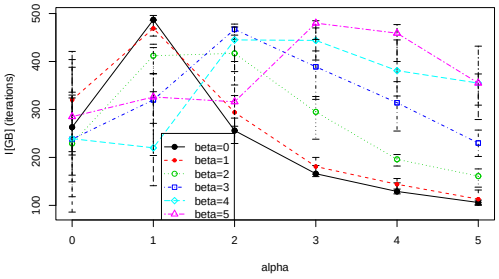
(c) P_{GB} using AS



(d) I_{GB} using AS

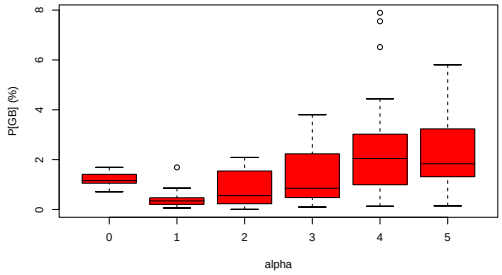


(e) P_{GB} using MMAS

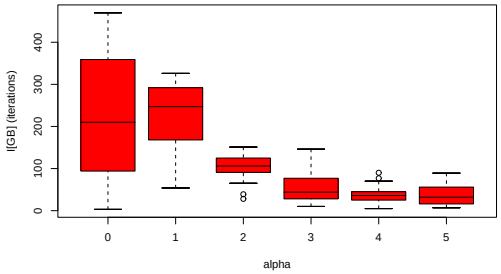


(f) I_{GB} using MMAS

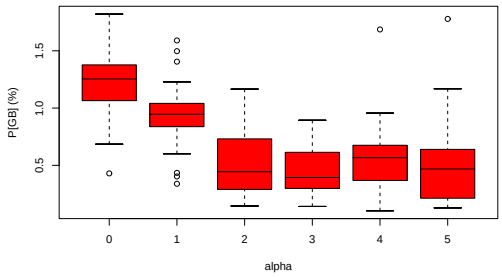
Figure 7.8: Example of the results achieved when varying α over GBAS, AS and MMAS on the TS data set using talent_10_150_4.



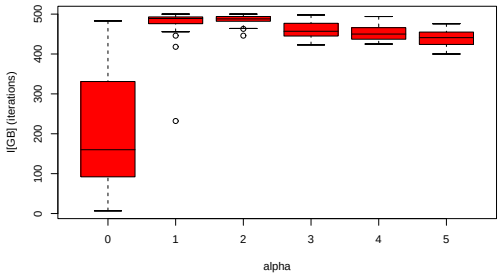
(a) P_{GB} using GBAS



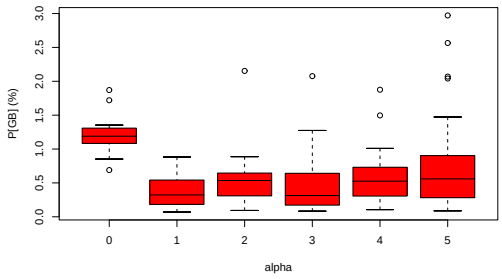
(b) I_{GB} using GBAS



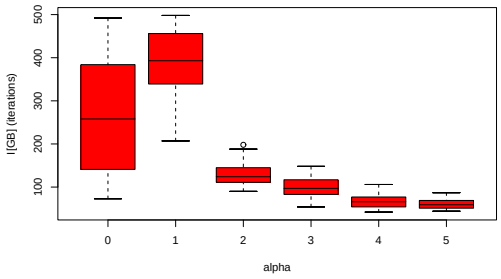
(c) P_{GB} using AS



(d) I_{GB} using AS

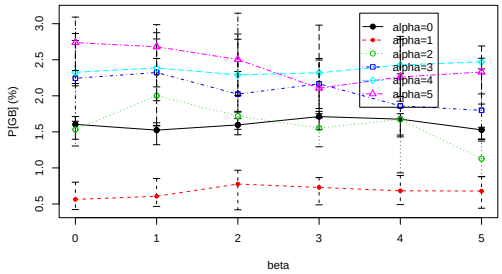


(e) P_{GB} using MMAS

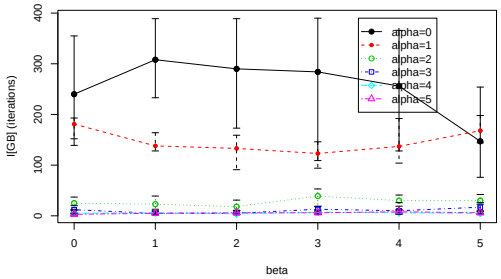


(f) I_{GB} using MMAS

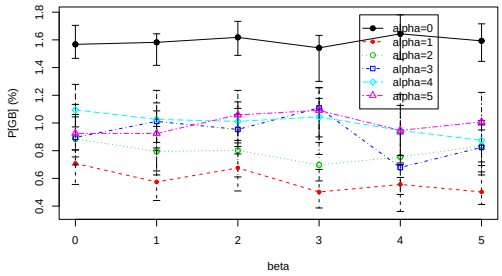
Figure 7.9: Example of the results achieved when varying α over GBAS, AS and MMAS on the QAP data set using tai60b.



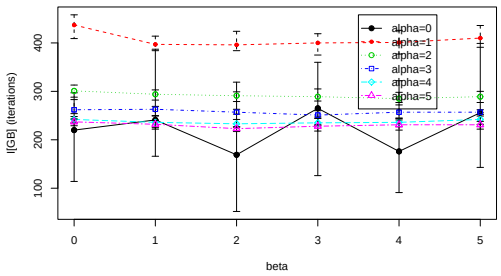
(a) P_{GB} using GBAS



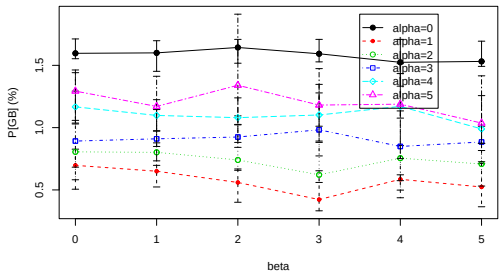
(b) I_{GB} using GBAS



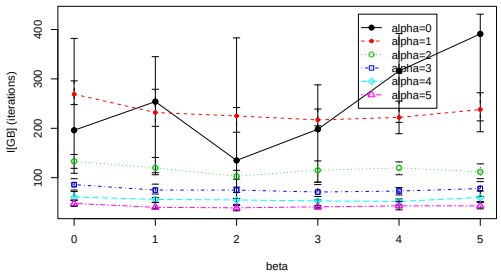
(c) P_{GB} using AS



(d) I_{GB} using AS

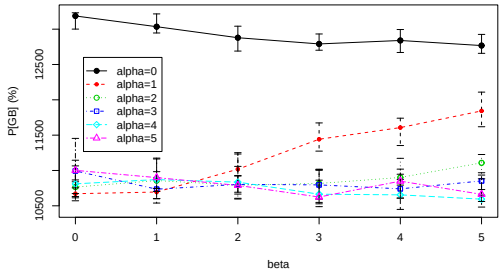


(e) P_{GB} using MMAS

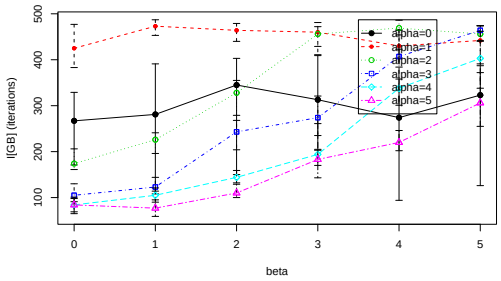


(f) I_{GB} using MMAS

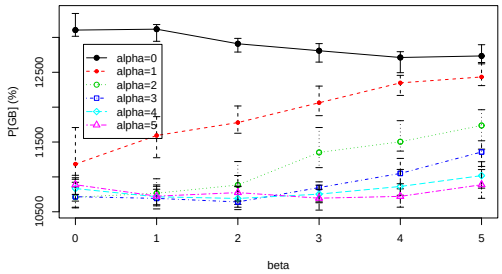
Figure 7.10: Example of the results achieved when varying β over GBAS, AS and MMAS on the TSP data set using att532.



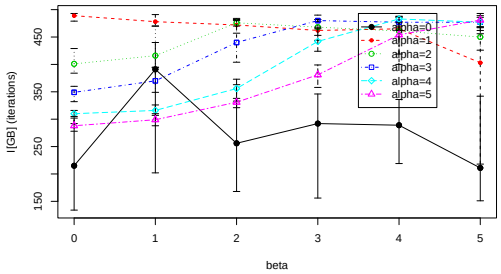
(a) P_{GB}



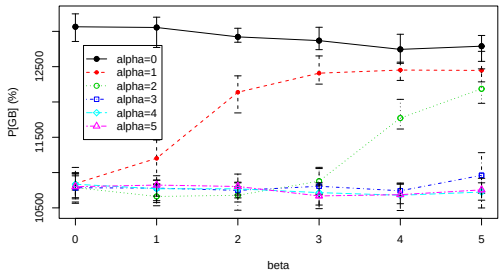
(b) I_{GB}



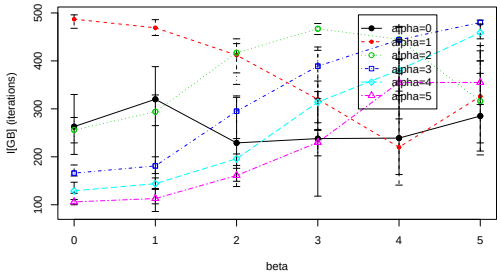
(c) P_{GB}



(d) I_{GB}



(e) P_{GB}



(f) I_{GB}

Figure 7.11: Example of the results achieved by varying β over GBAS, AS and MMAS on the TS data set using talent_10_150_4.

Problem	PRO(AS) P_{GB}	PRO(MMAS) P_{GB}	PRO(AS) I_{GB}	PRO(MMAS) I_{GB}
kroA150	19	31	31	66
rat195	33	36	58	75
gr202	28	28	25	64
att532	9	33	17	53
rat575	44	44	50	78
nug20	17	17	50	67
sko100a	17	17	17	100
lipa40a	17	17	17	33
tho150	17	17	17	33
esc16g	100	100	50	0
bur26d	33	33	0	83
tai35b	0	50	0	67
tai60b	17	17	17	100
talent_10_70_8	31	42	33	64
talent_10_100_5	36	44	28	53
talent_10_150_4	50	58	17	47
talent_10_175_4	39	50	8	44
talent_10_200_5	72	64	14	47

Table 7.9: Table showing how AS and MMAS rankings compare to GBAS for various α values.(PRO(X)=Percentage of points from the algorithm X that are in the same rank as in the corresponding GBAS runs)

Problem	PRO(AS) P_{GB}	PRO(MMAS) P_{GB}	PRO(AS) I_{GB}	PRO(MMAS) I_{GB}
kroA150	19	31	25	25
rat195	22	19	11	17
gr202	22	17	11	19
att532	3	11	11	28
rat575	22	33	11	22
talent_10_70_8	27	17	58	56
talent_10_100_5	31	31	67	58
talent_10_150_4	36	31	36	53
talent_10_175_4	50	25	42	67
talent_10_200_5	36	42	33	58

Table 7.10: Table showing how AS and MMAS rankings compare to GBAS for various β values. (PRO(X)=Percentage of points from the algorithm X that are in the same rank as in the corresponding GBAS runs)

Varying Parameter	Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
α	TSP	30	0	4
	QAP	8	1	2
	TS	30	30	24
β	TSP	30	3	2
	QAP	8	NA	NA
	TS	30	7	8

Table 7.11: Table summarising the hypothesis test results for P_{GB} varying α and β. (For more detailed results refer to Tables C.3, C.4, C.5 and C.6.)

Varying Parameter	Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
α	TSP	30	3	27
	QAP	8	1	7
	TS	30	6	15
β	TSP	30	3	4
	QAP	8	NA	NA
	TS	30	16	20

Table 7.12: Table summarising the hypothesis test results for I_{GB} varying α and β. (For more detailed results refer to Tables C.7, C.8, C.9 and C.10.)

7.5 Variation in *m*

The third experiment is to focus on the number of ants that the algorithm creates per iteration (m). There is no theoretical way to predict what the optimal value of m should be as it depends on an element of randomness and the complexity of the problem landscape. The question though is how do these algorithm react to varying this value and is there a point at which creating more solutions is no longer of value.

In the majority of work two views are taken. The first is that m should be equal to the number of components in a solution; for instance in the TSP the value should be the number of cities, n. The second view is that in fact very few ants are needed and the value is robust so that anything from 1 to 10 ants is enough. In Chapter 5 it was shown that without local search the latter view was the one that best described the behaviour of the algorithms, it is expected that local search will make this result more pronounced.

This section hopes to answer two questions, the first is to which view does GBAS subscribe, and the second is do Ant System and MMAS correlate to GBAS or not. These aspects are summarised in the following hypotheses, which can be replicated for MMAS and the other domains.

Null Hypothesis:	There is no significant correlation between GBAS and AS when varying m for P_{GB} for TSPs.
Alternative Hypothesis:	There is a significant correlation between GBAS and AS when varying m for P_{GB} for TSPs.

To answer this hypothesis this section will be split into two experiments. Firstly, each run will consist of m ants per iteration, and it will last for a predetermined number of iterations. The second experiment will run each algorithm for 1000 ants. The latter tries to capture the *efficacy* of the sampling being undertaken by the algorithm. Only at the end of the second experiment will these questions be able to be answered.

Varying the value of m was important to see how the algorithms performed with different amounts of sampling per update. The expectation was that the more ants that were used the better the quality of the solutions. It was varied as follows:

- for $n <= 50$, $m \in \{1, \ldots, n\}$,
- for $n > 50$, $m \in \{1, \ldots, 0.25n, 0.5n, n, 2n\}$.

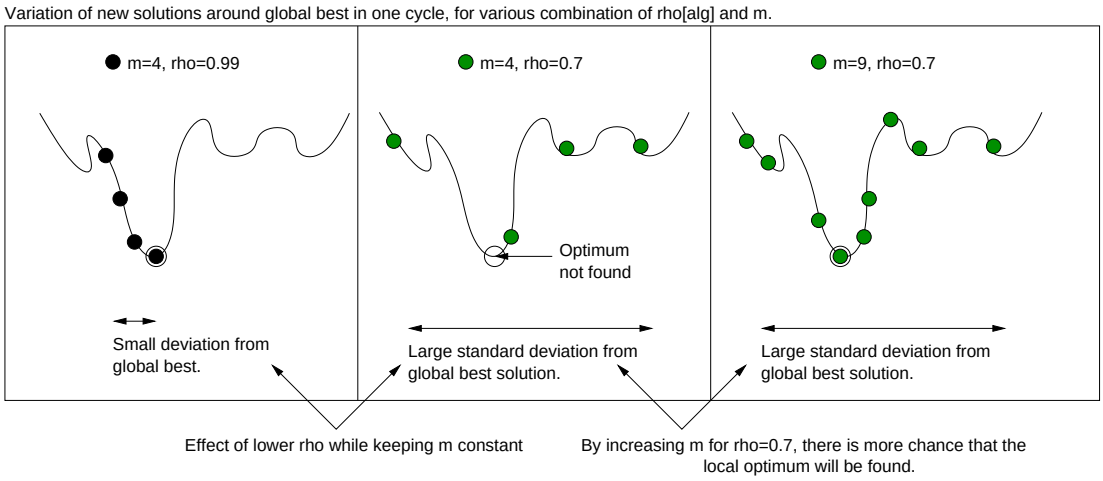


Figure 7.12: Figure illustrating why ρ would vary with m.

With the other parameters set as follows:

- $\alpha = 1$,
- $\beta = 0$,
- $\rho_{alg} \in \{0.7, 0.9, 0.95, 0.99\}$,
- $max_{it} = 500$.

ρ_{alg} was varied within a limited range, because m and ρ are assumed not to be independent. The variation was designed to show that large ρ_{alg} values would keep more information in the pheromone matrix, thus requiring less samples, and therefore a lower m, to move to a better solution. However, if m was large this would suit a lower ρ_{alg} value that would lose more of its information, thus require greater numbers of samples to maintain its search in the desired area. For instance, if one were to measure the standard deviation of solutions from the global best, to get the same number of solutions from the area around the global best when ρ_{alg} is set to a low value as when ρ_{alg} is set to a high value, the number of ants would have to be increased. Figure 7.12 gives a graphical representation of this. In Chapter 5 this was shown to be the case, and it is expected that local search will not change this behaviour.

Besides correlations, the results were viewed with respect to a set of four criteria:

- the ranking of ρ_{alg} for P_{GB} when $m = n$,
- the ranking of ρ_{alg} for I_{GB} when $m = n$,
- predictability of the rankings of ρ_{alg} as m varies,

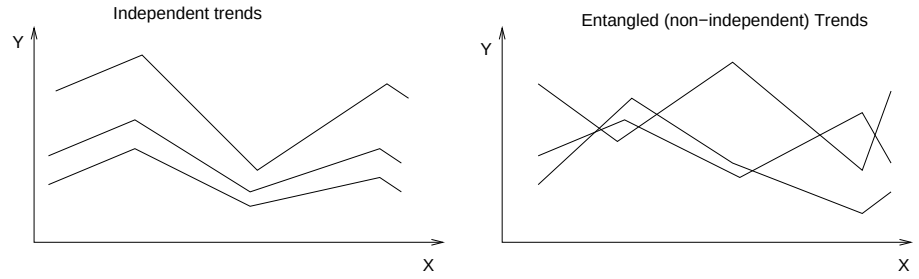


Figure 7.13: Figure showing independent and non-independent trends. These are the two cases the ranking of ρ_{alg} will identify.

- agreement of AS and MMAS rankings with the rankings of GBAS.

The first two criteria were to test the performance of each ρ_{alg} for this value of m to set a standard ranking for the algorithm. The value of n was chosen as this is the most commonly quoted value for experiments to use. The third criteria is a measure of how well the ranks of ρ_{alg} values for P_{GB} and I_{GB} at $m = n$ hold for the other m values. These results will be given as a percentage of the total number of points tested as in Equation 5.11.

$$\begin{aligned} &\text{Prediction(algorithm)} = \frac{\sum_{m=1}^{m=2n} O(m)}{\text{number of points}} * \frac{100}{1} \\ &\text{where } O(i) \begin{cases} 1 & \text{if ranks are the same,} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{7.3}$$

Finally, it is of interest to compare the orderings for AS and MMAS to those of GBAS and also the ordering that is expected of results taken for various ρ_{alg}, in this case $(0.7, 0.9, 0.95, 0.99)$. These were done to get a measure of the independence of each run. For instance, $\rho_{alg} = 0.7$ may begin the worst setting for this parameter but then as m gets larger it becomes better than the other settings for ρ_{alg}(see Figure 7.13).

The expected outcome from GBAS is that as ρ_{alg} tends to 1, the performance will improve, as well as an improvement as m tends to ∞.

Assumption: the problems are not deceptive and therefore an increase in knowledge about previous solutions is helpful with respect to the current search window.

The expected correlations are expected to be the same as those in Chapter 5, which are as follows:

- For P_{GB}, GBAS and AS are significantly correlated, reject Null Hypothesis.

- For P_{GB}, GBAS and MMAS are significantly correlated, reject Null Hypothesis.
- For I_{GB}, GBAS and AS are not significantly correlated, reject Alternative Hypothesis. Most likely to be correlated for $\rho_{alg} \in \{0.7, 0.9\}$.
- For I_{GB}, GBAS and MMAS are not significantly correlated, reject Alternative Hypothesis.

Figure 7.14 gives an example from the results of the TSP set of problems. For all the algorithms it can be observed that the performance variable P_{GB} is relatively constant across m values in the range $[1, 20]$. As m increases above 20, a decrease in the performance occurs, which is due to the lack of iterations able to be processed for these larger problems in the allotted time. The increase is not very large and one can assume that the performance would continue to be relatively constant if the full 500 iterations were able to be performed. The best value of ρ_{alg} appears to be 0.95 for all three algorithm. However, 0.99 did perform better for GBAS than for either of the other two algorithms, a consequence having the largest value for I_{GB}.

For the QAP in Figure 7.15, the results are similar with the curves being relatively flat across various values of m. For GBAS, when n is small ρ_{alg} tends to be best at 0.95 but as n rises 0.9 and 0.95 interchange. For AS and MMAS the best value is 0.9, although 0.95 delays convergence for the longest time. This is another indication that later convergence does not necessary lead to better performance.

In Figure 7.16 are graphs for the Talent Scheduling problems. These show that there is little difference in performance for GBAS between 0.7 and 0.95. However, AS and MMAS prefer values of 0.7 and 0.9 for performance even with 0.95 offering the best I_{GB} value.

Across all the problems the performance has not increased with m, therefore a value of $m = 1$ seems a reasonable value to consider. From Tables 7.13 and 7.14 it would appear that GBAS adheres the best to the expected ordering $(0.7, 0.9, 0.95, 0.99)$, with the TSP and QAP showing a greater adherence compared to the results of the Talent Scheduling problems.

Table 7.15 shows that for GBAS the rankings of the various ρ_{alg} values when $m = n$ for most of the TSP and QAP problems behave as expected. For these two domains there is some movement between the values in the range of 0.9 to 0.99, but most of the problems show 0.7 to be least favourable setting. Moving across to Ant System

and then on to MMAS, the best setting moves from 0.99 to 0.95, with the former becoming the worst in many problems. All three algorithms performed best on the Talent Scheduling problems using a setting less than 0.99.

For Table 7.16, GBAS shows the most regular and predictable rankings, with the latest convergence being achieved as expected with $\rho_{alg} = 0.99$. Ant System prefers a value of 0.95, and MMAS is more problem dependent. The same tables in Section 5.6 show that the rankings, for both P_{GB} and I_{GB}, are much more predictable for all the algorithms when local search is not used.

The stability of the rankings from $m = n$ across a range of values is illustrated in Table 7.17. This table shows that the algorithms show different levels of stability in the order $GBAS > AS > MMAS$. From these results it seems that the rankings for I_{GB} hold their positions, across a range of m values, better than P_{GB}. These results concur with those from Table 5.12 when the algorithms were run without local search.

In Table 7.18, MMAS appears to be more correlated with GBAS than AS is for I_{GB}, and both are fairly even for P_{GB}. When compared to Table 5.15 it seems that the rankings, for both AS and MMAS, are more similar to those of GBAS with local search, than when there was none. This might indicate that the local search method may make the algorithms more alike in terms of performance and convergence, therefore making the differences in structures such as the pheromone matrix less critical.

In Table 7.19, the correlations for the variable P_{GB} between the three algorithms are given. One can see it is very different from the correlation tables in Chapter 5 which showed 100% significant correlations for both AS and MMAS. For the TSP, both algorithms prefer problems with large n, although there are scattered significant correlations in the smaller problems. The two other domains are a complete contrast to when no local search was used. For QAP and Talent Scheduling the results show almost no significant correlations at all.

Tables 7.20 show the correlations for variable I_{GB}. The TSP problems are significantly correlated for both algorithms in a similar manner to when local search was not used. The Talent Scheduling problems has become more correlated with the addition of the local search method. In contrast, for the QAP, the significant correlations are less numerous now than without local search.

The correlations allow the following conclusions to be drawn:

- For P_{GB},TSP: GBAS and AS are significantly correlated, reject Null Hypothesis.
- For P_{GB},QAP: GBAS and AS are not significantly correlated, reject Alternative Hypothesis.
- For P_{GB},TS: GBAS and AS are not significantly correlated, reject Alternative Hypothesis. As n increases so correlations become more frequent.
- For P_{GB},TSP: GBAS and MMAS are significantly correlated, reject Null Hypothesis.
- For P_{GB},QAP: GBAS and MMAS are not significantly correlated, reject Alternative Hypothesis.
- For P_{GB},TS: GBAS and MMAS are not significantly correlated, reject Alternative Hypothesis. As n increases so correlations become more frequent.
- For I_{GB},TSP: GBAS and AS are significantly correlated, reject Null Hypothesis.
- For I_{GB},QAP: GBAS and AS are not significantly correlated, reject Alternative Hypothesis.
- For I_{GB},TS: GBAS and AS are significantly correlated, reject Null Hypothesis.
- For I_{GB},TSP: GBAS and MMAS are significantly correlated, reject Null Hypothesis.
- For I_{GB},QAP: GBAS and MMAS are not significantly correlated, reject Alternative Hypothesis.
- For I_{GB},TS: GBAS and MMAS are significantly correlated, reject Null Hypothesis.

Comparing these results to those in Chapter 5, it is immediately clear that the distinction between GBAS and the other two algorithms is more problem dependent. This is likely a result of adding a problem specific local search method. When no local search was used GBAS and AS were significantly correlated for P_{GB}, now they are not, with the exception of the TSP, which has fewer significant correlations but maintains a majority. Although significant correlations are still possible their frequency is now in the minority.

For the TSP, MMAS is still significantly correlated with GBAS, although it is not 100% and therefore depends on the problem, for instance gr202 is not significantly correlated for the majority of ρ_{alg} values. For the QAP and TS there is no longer a clear significant

Algorithm	Min.	Q1	Median	Mean	S.d.	Q3	Max.
GBAS	0.00	0.00	0.00	33.60	41.72	78.27	95.83
(TSP)	0.00	20.83	91.67	60.83	46.64	95.83	95.83
(QAP)	0.00	0.00	33.75	37.57	40.36	74.81	80.00
(TS)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AS	0.00	0.00	0.00	1.33	4.75	0.00	20.00
(TSP)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
(QAP)	0.00	0.00	0.00	2.98	7.01	0.96	20.00
(TS)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MMAS	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 7.13: Table showing how, for each algorithm, the ranking of ρ_{alg} compares to the expected ordering (0.7,0.9,0.95,0.99) for P_{GB}. Statistics have been broken down by problem type where necessary.

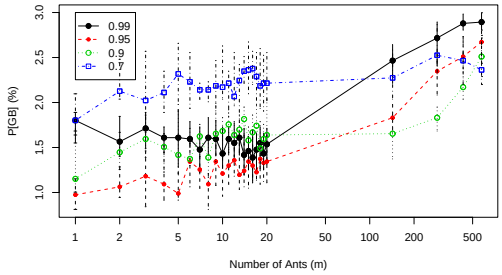
Algorithm	Min.	Q1	Median	Mean	S.d.	Q3	Max.
GBAS	0.00	64.58	100.00	74.31	40.58	100.00	100.00
(TSP)	83.33	83.33	100.00	93.33	9.13	100.00	100.00
(QAP)	0.00	77.08	100.00	76.04	44.41	100.00	100.00
(TS)	0.00	4.17	58.33	52.50	49.09	100.00	100.00
AS	0.00	0.00	0.00	5.69	15.61	0.00	65.00
(TSP)	0.00	0.00	0.00	5.00	7.45	8.33	16.67
(QAP)	0.00	0.00	0.00	9.69	22.77	3.13	65.00
(TS)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MMAS	0.00	0.00	21.96	28.19	30.62	43.12	87.50
(TSP)	16.67	29.17	83.33	60.00	34.18	83.33	87.50
(QAP)	0.00	0.00	21.96	19.69	18.32	31.25	48.57
(TS)	0.00	0.00	0.00	10.00	20.11	4.17	45.83

Table 7.14: Table showing how, for each algorithm, the rankings of ρ_{alg} compares to the expected ordering (0.7,0.9,0.95,0.99) for I_{GB}. Statistics have been broken down by problem type where necessary.

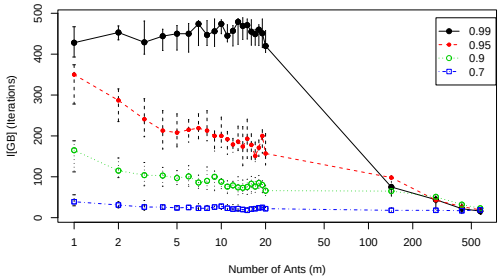
correlation between the two algorithms, although as *n* increases significant correlations do become more frequent.

In terms of I_{GB}, the results are also more complex. For the TSP, there was no significant correlations between AS and GBAS when no local search was used, but there is now a majority of significant correlations. This is also the same for Talent Scheduling. QAP is still mixed and therefore agrees with the rejecting of the Alternative Hypothesis result reached in Chapter 5. The same results apply to MMAS and GBAS.

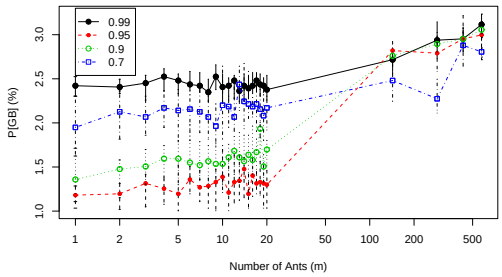
These results show that when local search is added the situation becomes fuzzy and less distinct than was found in Chapter 5 when no local search method was used. It is interesting to see that the significant correlations have decreased for P_{GB}, but increased for I_{GB}. This indicates that the local search method helps to standardise the convergence of the algorithms, while each algorithm is able to take advantage of the local search to different degrees. This is most likely to do with the structure of the pheromone matrices and how varied the solutions are that are being constructed from these matrices.



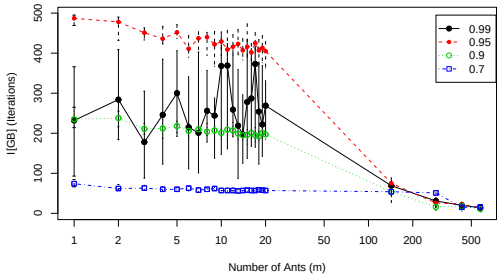
(a) P_{GB} using GBAS



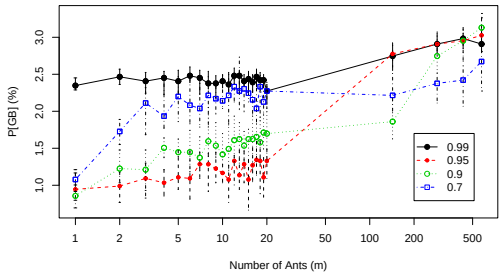
(b) I_{GB} using GBAS



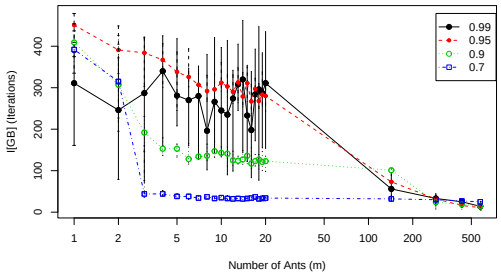
(c) P_{GB} using AS



(d) I_{GB} using AS

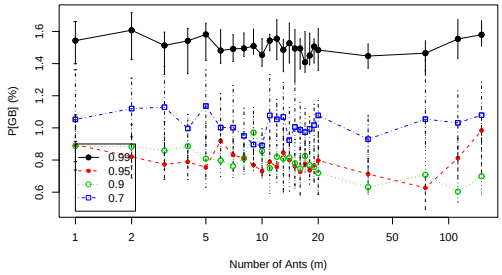


(e) P_{GB} using MMAS

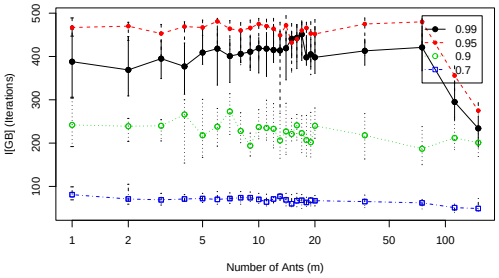


(f) I_{GB} using MMAS

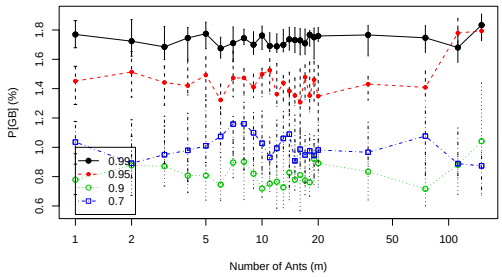
Figure 7.14: Example of the results achieved by varying ρ_{alg} with m over GBAS, AS and MMAS, in this case for the TSP problem rat575. The x-axis is a log scale.



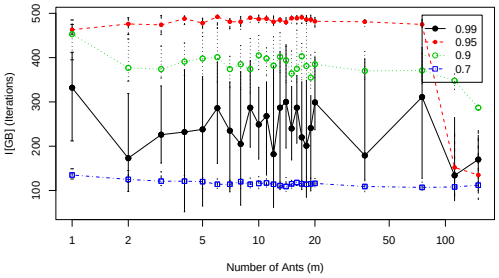
(a) P_{GB} using GBAS



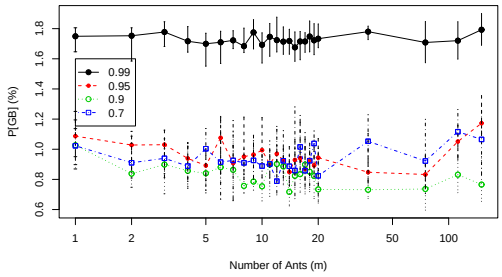
(b) I_{GB} using GBAS



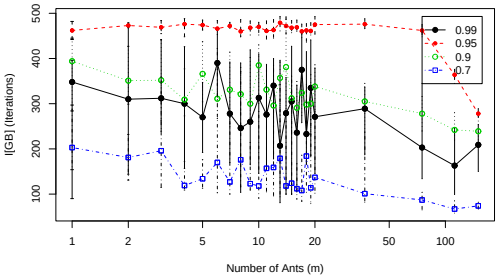
(c) P_{GB} using AS



(d) I_{GB} using AS

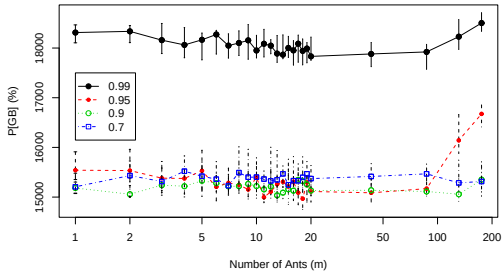


(e) P_{GB} using MMAS

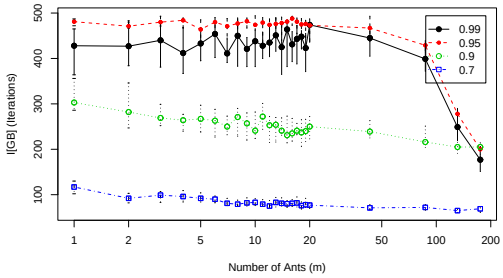


(f) I_{GB} using MMAS

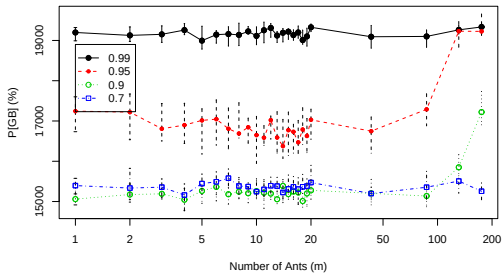
Figure 7.15: Example of the results achieved by varying ρ_{alg} with m over GBAS, AS and MMAS, in this case for the QAP problem tho150. The x-axis is a log scale.



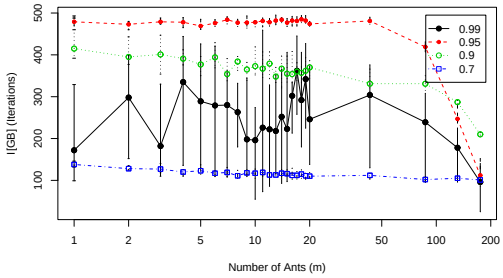
(a) P_{GB} using GBAS



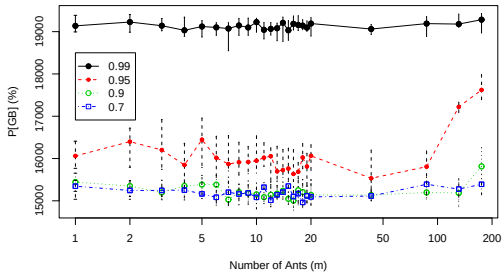
(b) I_{GB} using GBAS



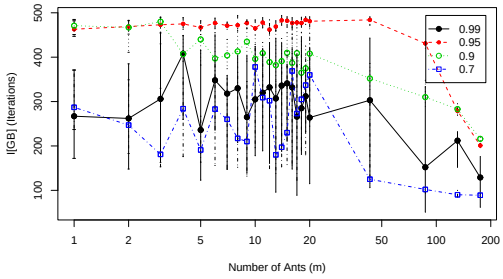
(c) P_{GB} using AS



(d) I_{GB} using AS



(e) P_{GB} using MMAS



(f) I_{GB} using MMAS

Figure 7.16: Example of the results achieved by varying ρ_{alg} with m over GBAS, AS and MMAS, in this case for the TS problem talent_10_175_4. The x-axis is a log scale.

Problem	GBAS				AS				MMAS			
	0.7	0.9	0.95	0.99	0.7	0.9	0.95	0.99	0.7	0.9	0.95	0.99
kroA150	4	3	2	1	4	2	1	3	4	3	1	1
rat195	4	3	2	1	4	2	1	3	4	2	1	2
gr202	4	3	2	1	4	2	1	3	4	2	1	3
att532	4	1	2	3	3	2	4	1	1	2	3	4
rat575	2	1	3	4	1	2	2	2	1	2	2	4
nug20	4	3	2	1	4	3	1	1	3	3	2	1
sko100a	3	2	1	4	3	1	2	4	3	2	1	4
lipa40a	4	3	2	1	4	3	1	2	2	2	1	4
tho150	3	1	2	4	2	1	4	3	3	1	2	4
esc16g	1	1	1	1	1	1	1	1	1	1	1	1
bur26d	4	3	2	1	4	3	1	2	3	1	1	3
tai35b	4	3	2	1	3	2	1	4	3	2	1	4
tai60b	4	2	1	3	2	1	3	4	2	3	1	4
talent_10_70_8	4	1	2	3	3	2	1	4	3	1	2	4
talent_10_100_5	3	2	1	4	3	2	1	4	3	1	2	4
talent_10_150_4	3	2	1	4	2	1	3	4	3	1	2	4
talent_10_175_4	2	1	3	4	1	2	3	4	2	1	3	4
talent_10_200_5	1	2	3	4	1	2	3	4	1	2	3	4

Table 7.15: Table showing rankings of ρ_{alg} for P_{GB} when $m = n$. (1=closest to optimal)

Problem	GBAS				AS				MMAS			
	0.7	0.9	0.95	0.99	0.7	0.9	0.95	0.99	0.7	0.9	0.95	0.99
kroA150	1	2	3	4	1	2	4	3	1	2	3	4
rat195	1	2	3	4	1	2	4	3	1	2	3	4
gr202	1	2	3	4	1	2	4	3	1	2	3	4
att532	1	4	3	2	2	1	3	3	4	2	3	1
rat575	1	4	3	2	1	2	4	3	4	2	1	3
nug20	1	2	3	4	1	2	3	4	1	2	3	4
sko100a	1	2	3	4	1	3	4	2	1	3	4	2
lipa40a	1	2	3	4	1	2	4	3	1	2	3	4
tho150	1	2	4	3	1	4	3	2	1	3	4	2
esc16g	1	4	1	3	1	1	4	1	1	1	3	3
bur26d	1	2	3	4	1	2	4	3	3	1	2	4
tai35b	1	2	3	4	1	3	4	2	2	1	3	4
tai60b	1	2	3	4	1	3	4	2	1	2	3	4
talent_10_70_8	1	2	3	4	1	2	4	3	2	1	4	3
talent_10_100_5	1	2	3	4	1	3	4	2	1	2	3	4
talent_10_150_4	1	2	3	4	1	3	4	2	1	2	4	3
talent_10_175_4	1	2	4	3	1	4	3	2	1	4	3	2
talent_10_200_5	1	4	3	2	3	4	1	2	1	4	3	2

Table 7.16: Table showing rankings of ρ_{alg} for I_{GB} when $m = n$. (4=latest convergence)

Problem	P_{GB} Stability			I_{GB} Stability		
	GBAS	AS	MMAS	GBAS	AS	MMAS
kroA150	96	42	21	100	79	83
rat195	92	92	13	100	92	88
gr202	96	54	58	100	92	83
att532	4	4	8	8	4	4
rat575	4	4	4	4	75	4
nug20	80	25	10	100	65	30
sko100a	71	75	13	100	71	42
lipa40a	68	8	5	100	88	35
tho150	46	4	38	92	4	83
esc16g	100	100	100	6	19	13
bur26d	73	58	4	100	100	15
tai35b	80	49	43	100	60	31
tai60b	50	88	17	100	92	20
talent_10_70_8	8	63	21	100	92	25
talent_10_100_5	67	42	25	100	54	4
talent_10_150_4	54	100	4	58	92	17
talent_10_175_4	25	21	33	92	4	13
talent_10_200_5	17	17	42	8	4	17

Table 7.17: Table showing the stability of the rankings when using GBAS, AS and MMAS. (Percentages are given as integers.)

Problem	PRO(AS) P_{GB}	PRO(MMAS) P_{GB}	PRO(AS) I_{GB}	PRO(MMAS) I_{GB}
kroA150	34	43	57	86
rat195	27	25	54	90
gr202	16	21	48	90
att532	20	19	39	57
rat575	35	32	46	48
nug20	54	28	83	56
sko100a	51	54	32	34
lipa40a	25	14	56	64
tho150	35	52	47	56
esc16g	100	100	41	34
bur26d	47	13	50	56
tai35b	11	18	35	72
tai60b	4	19	27	45
talent_10_70_8	61	64	48	68
talent_10_100_5	56	45	36	38
talent_10_150_4	39	41	38	31
talent_10_175_4	54	49	50	43
talent_10_200_5	68	58	46	41
Mean	43	41	50	59

Table 7.18: Table showing the similarity of the ranks of the points between AS,MMAS and GBAS. (Cor=Pearson Product-Moment Correlation, PRO(X)=Percentage of points from the running of algorithm X that are in the same rank as in the corresponding GBAS runs given as integers.)

Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
TSP	20	11	14
QAP	32	7	8
TS	20	7	8

Table 7.19: Table summarising the hypothesis test results for P_{GB} varying m. (For more detailed results refer to Tables C.11 and C.12.)

Domain	Number of Problems	AS,GBAS Significant	MMAS,GBAS Significant
TSP	20	15	17
QAP	32	14	10
TS	20	17	15

Table 7.20: Table summarising the hypothesis test results for I_{GB} varying m. (For more detailed results refer to Tables C.13 and C.14.)

7.6 Variation in m_2

When considering the question of how many ants is a good number to assign to the variable m, besides the direct experiment it is also important to consider how much useful work is being done by those ants, known as *efficacy*. This is very important as the construction method used to build the solutions takes longer than other algorithms, such as Tabu Search or Simulated Annealing.

Therefore in this section the number of samples is limited to 1000 and the number of iterations is determined by Equation 7.4.

$$1000 = max_{it} \cdot m \tag{7.4}$$

The reasons for picking the number of 1000 was described in Chapter 5. This value was not altered to accommodate the larger problem sizes to allow comparisons to be made with the experiments in Chapter 5, but it still provides a valid measure as it is common across problems and algorithms.

As for the previous experiment, the following conditions held for all experiments.

- $\alpha = 1$
- $\beta = 0$
- $\rho_{alg} \in \{0.7, 0.9, 0.95, 0.99\}$

For each test m was varied as follows:

- for $n <= 50$, $m \in \{1, \ldots, n\}$
- for $n > 50$, $m \in \{1, \ldots, 20, 0.25n, 0.5n, 0.75n, n\}$.

In Figure 7.17 an example of the TSP results is given for each algorithm. One immediately observes the fact that for all the algorithms the best results can be achieved for $m = 1$ for all values of ρ_{alg}. Focusing on GBAS, the normal ranking of $0.7 > 0.9 > 0.95 > 0.99$ exists until m increases to around $m = 6$, when 0.9 and 0.95 become better values for ρ_{alg}. For the variable I_{GB} the value of 0.99 provides over twice as many iterations as the other values while m is small; as m increases, I_{GB} reduces until it is at the same level as the other ρ_{alg} settings. Below this setting of ρ_{alg}, the I_{GB} value changes little in the period where $m < 20$. This is most likely because the problems are so big that whether you use 1 or 20 ants becomes irrelevant.

For AS and MMAS the incline as m increases is steeper than for GBAS. As expected 0.95 is the preferred ρ_{alg} value for P_{GB} in both algorithms. In contrast to the other two algorithms, MMAS is almost ρ_{alg} agnostic when it is set below 0.99 for small m. This makes for the interesting situation that for MMAS good solutions within 1% of the optimum can be reached with $m = 1$, for any ρ_{alg}.

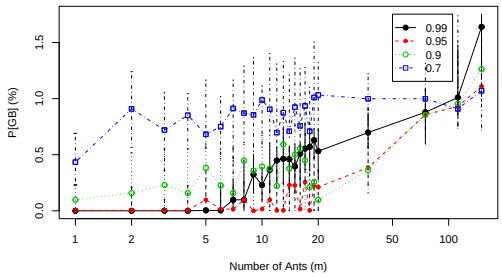
For the QAP graphs in Figure 7.18 similar trends are shown, although none of the algorithms are as flat as their TSP counterparts while m is small. The most varied results come from the Talent Scheduling results in Figure 7.19. The graphs are noticeably more like those for runs that do not use a local search method. However, the best results could still be achieved by all algorithms with only one ant and with $\rho_{alg} < 0.99$. Across all the domains it is striking how similar all the algorithms are and that the scale of any differences are much smaller than without the local search method.

In Figures 7.20, 7.21 and 7.22, the plots of the number of ants (m) against the number of components (n) are shown. It is clear that $m = 1$ is the preferable setting for all the algorithms, for all problems tested, for $\rho_{alg} = 0.99$. Below this settings m varies up to as much as 20 for some domains.

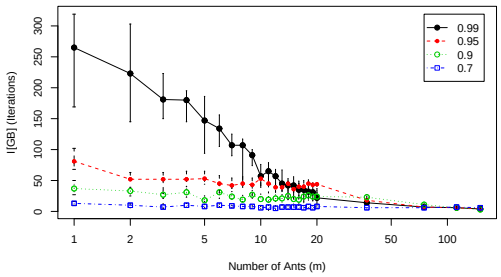
For the TSP, GBAS and AS, while ρ_{alg} was low, m can be increased to as much as 8 ants; although as n gets larger this figure diminishes. For the QAP, GBAS and AS show great variation for smaller n, reaching to 20 ants for some small problems. This is indicative of the QAP difficulty, but for $\rho_{alg} > 0.7$ the number of ants needed is less than or equal to 10, which is similar for the results with no local search. Finally for the Talent Scheduling problem, there is variation in the values m takes. The trend is as expected as $\rho_{alg} \to 0$ the more ants are required, although 10 ants is sufficient for most problems. For all the domains, MMAS is the algorithm that uses the lowest numbers of ants, which may indicate that MMAS makes better use of the solutions created by the ants.

From these results it can be concluded that the view that few ants should be used for the best results seems to be validated. However, the fact that all the algorithms, both with local search and without local search, perform so well with only a single ant is curious. For an algorithm that is suppose to thrive on its diversity and sampling, the question has to be asked "how does a single ant sample a given space?" Therefore the other question that has to be asked is, given that no population is actually used, are these Ant algorithms built for use with Combinatorial Optimisation Problems that are

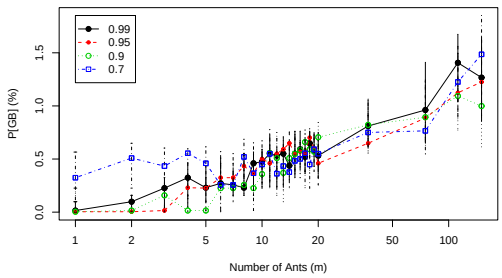
not heavily distributed and require a degree of parallel computation?



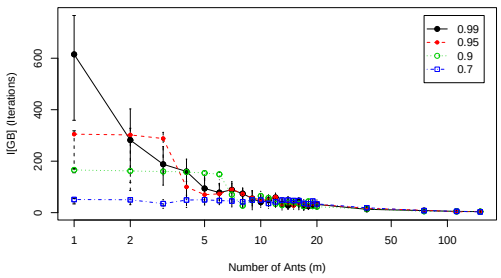
(a) P_{GB} using GBAS



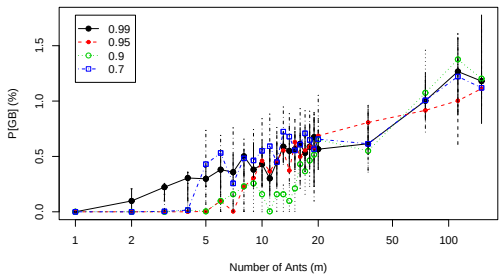
(b) I_{GB} using GBAS



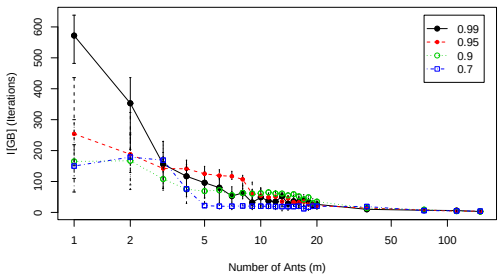
(c) P_{GB} using AS



(d) I_{GB} using AS

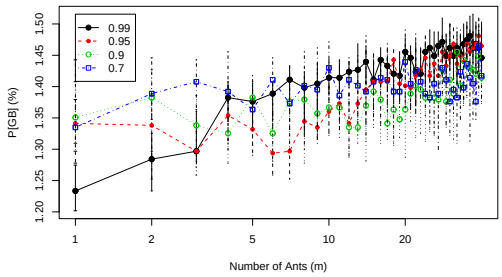


(e) P_{GB} using MMAS

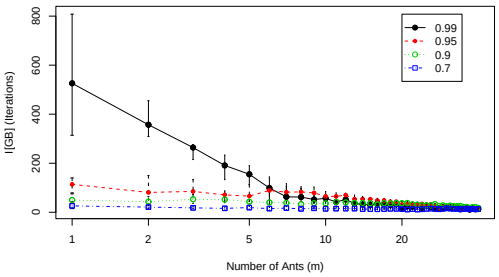


(f) I_{GB} using MMAS

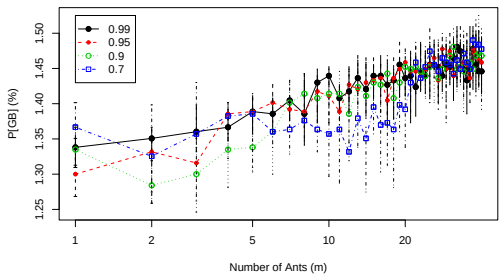
Figure 7.17: Example of the results achieved varying m, but limiting the runs to a maximum of 1000 samples, over GBAS, AS and MMAS on the TSP data set, in this case for kroA150. The x-axis is a log scale.



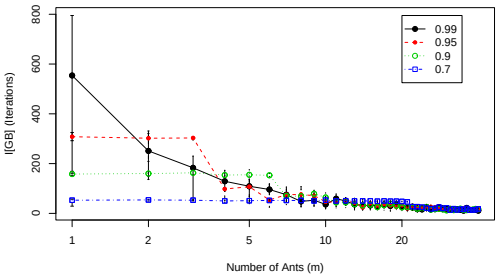
(a) P_{GB} using GBAS



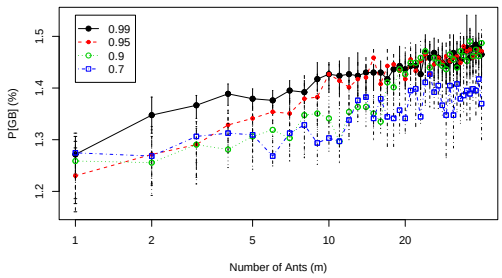
(b) I_{GB} using GBAS



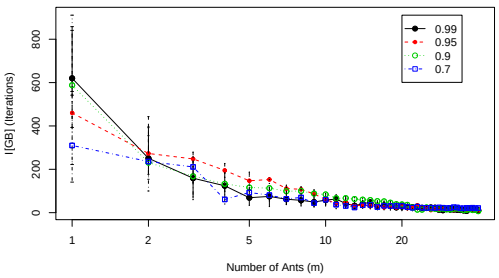
(c) P_{GB} using AS



(d) I_{GB} using AS

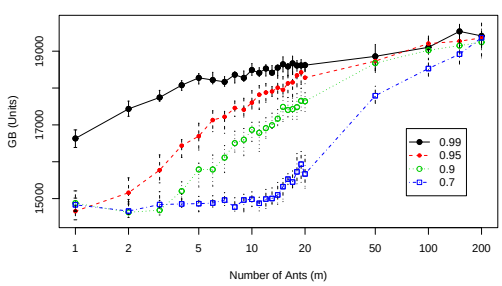


(e) P_{GB} using MMAS

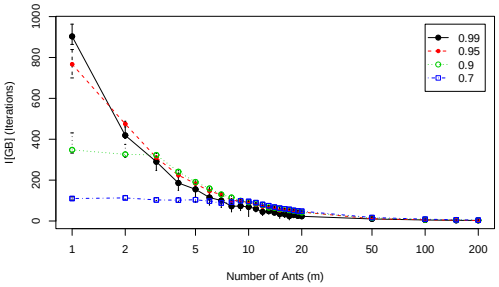


(f) I_{GB} using MMAS

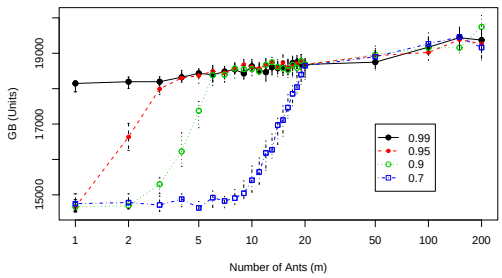
Figure 7.18: Example of the results achieved varying m, but limiting the runs to a maximum of 1000 samples, over GBAS, AS and MMAS on the QAP data set, in this case for lipa40a. The x-axis is a log scale.



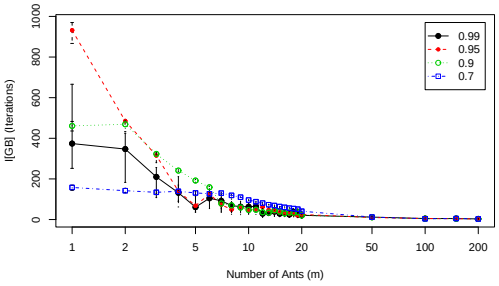
(a) P_{GB} using GBAS



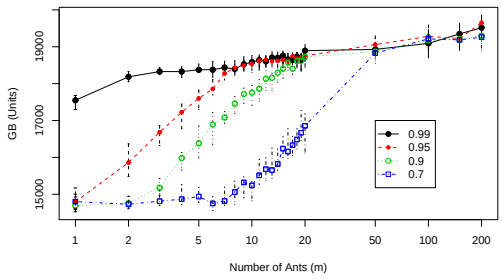
(b) I_{GB} using GBAS



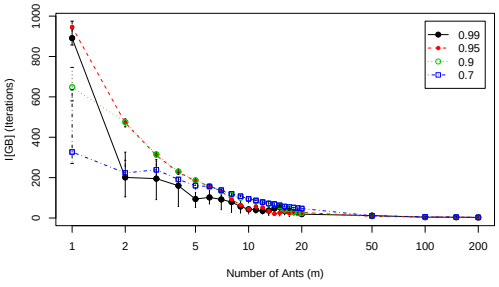
(c) P_{GB} using AS



(d) I_{GB} using AS

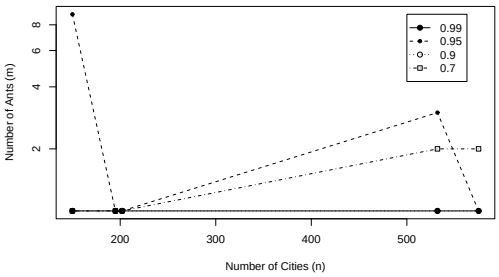


(e) P_{GB} using MMAS

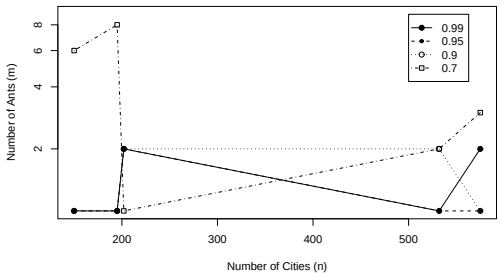


(f) I_{GB} using MMAS

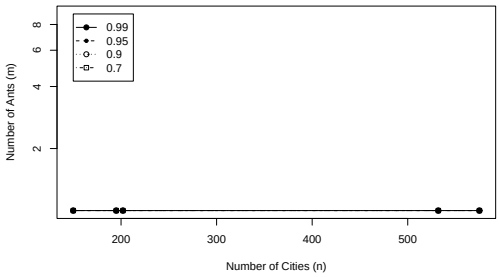
Figure 7.19: Example of the results achieved varying m, but limiting the runs to a maximum of 1000 samples, over GBAS, AS and MMAS on the TS data set, in this case for talent_10_200_5. The x-axis is a log scale.



(a) TSP using GBAS

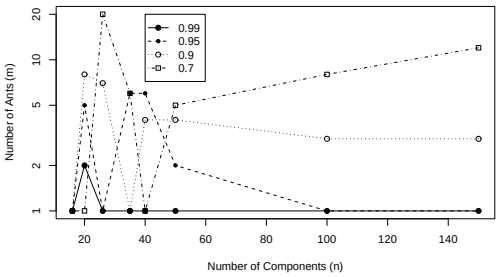


(b) TSP using AS

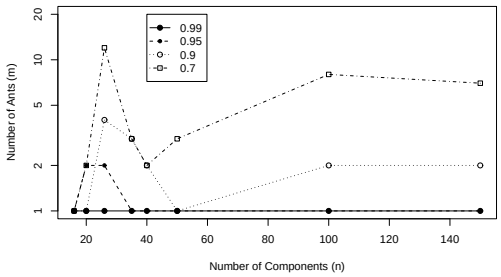


(c) TSP using MMAS

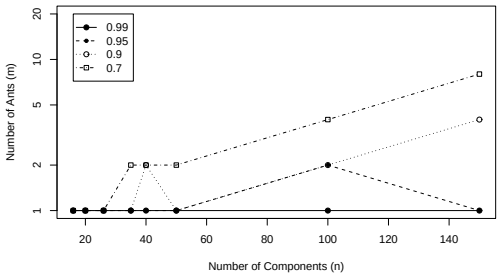
Figure 7.20: Figures for the TSP data set showing how the optimum m varies with the number of cities, plotted on log(y)-x axes.



(a) QAP using GBAS

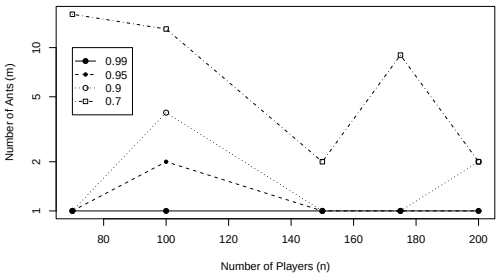


(b) QAP using AS

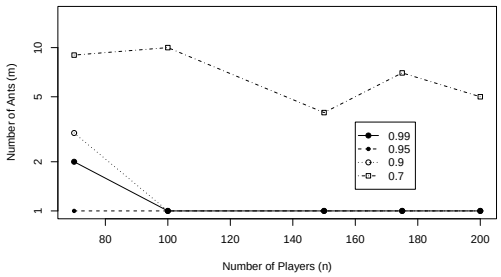


(c) QAP using MMAS

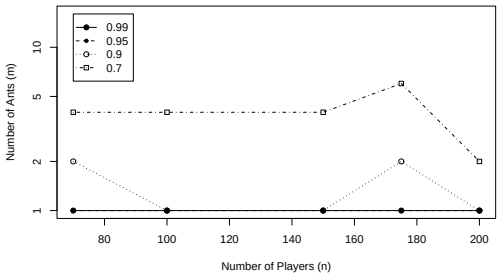
Figure 7.21: Figures for the QAP data set showing how the optimum m varies with the number of components in the problem, plotted on log(y)-x axes.



(a) TS using GBAS



(b) TS using AS



(c) TS using MMAS

Figure 7.22: Figures for the TS data set showing how the optimum m varies with the number of pieces, plotted on log(y)-x axes.

7.7 Conclusion

Null Hypothesis:	Local search does not significantly alter the way the the parameters affect the performance of the algorithms.
Alternative Hypothesis:	Local search does significantly alter the way the parameters affect the performance of the algorithms.

In terms of the original hypothesis, shown above, this chapter gives evidence that the Alternative Hypothesis is true. If the correlations found in Chapter 5 were not reversed, then the conclusions were made much more problem and domain dependent.

In Section 7.3 the parameter ρ_{alg} was varied, and it was shown that relationships existed between GBAS and both AS and MMAS, which had not existed when no local search methods had been used. Without local search the results were distinct and did not vary between problems or domains. In contrast, when a local search method was added, although most of the problems were correlated there was a large minority that did not show significant correlation.

In Section 7.4, the parameters α and β were varied and it was shown that not only does adding a local search method have an effect, but the strength of the local search method matters as well. If you include a local search method that is very strong, then there is no longer any correlation between increasing the input of the heuristic and the quality of the final solution. For both these parameters it was found that over 50% of the relationships that emerged in Chapter 5 no longer are true.

In Section 7.5, the parameter m was varied alongside ρ_{alg}. In this section it was shown that the conclusions became more complex and that hypotheses could only be applied to specific domains and varied greatly between the problems within these. The evidence indicated that the presence of local search, while on the one hand making the algorithm more specific to the problem, also minimised the difference between the algorithms in terms of their convergence.

Finally in Section 7.6 the parameter m was varied, but for a set number of samples. This showed that all the algorithms could solve many of the problems with only a single ant. This raises the question of how this algorithm is sampling the search space. The reason why one ant is so effective is because local search is used on every ant produced. This means that the variation in solutions that seed the local search is maximised, enabling

the local search to find the most local optima.

In the future it would be of interest to increase the number of problems, especially get more QAP problems above 100 locations in size. It would also be of interest to vary the local search policy to using the method on all solutions, rather than just the best one per iteration, although considering the best performance is gained with a single ant this may not yield anything new apart from better quality solutions.

Given these changes that local search seems to have on the Ant algorithms, in the next chapter the question is asked “Are Ant algorithms really suited to driving a local search method?”

7.8 Summary

In this chapter local search methods were added to various Ant algorithms and it was shown that this did have an effect on the parameters and how the algorithms correlated to GBAS. In the next chapter the case for using Ant algorithms to drive local search methods for Combinatorial Optimisation problems is focused on.

Chapter 8

Understanding the role of Ant Algorithms when combined with Local Search

Having seen in the previous chapter that a local search method can change how an Ant algorithm behaves, this chapter addresses the question "how does an Ant algorithm interact with a local search method?" The chapter questions previous assumptions which were based on how Ant algorithms work without local search and shows that it is possible that a better method for driving the local search could be found.

The chapter begins with an introduction to how Ant algorithms are thought to work with local search. Afterwards there is some discussion on how earlier studies have tried to show that the pheromone matrix in its current form is a necessary component in the production of good quality solutions. In Section 8.2 the concept of a *Shaking* algorithm is introduced and a specific implementation is described. Section 8.3 describes the method that was used to gain the results discussed in Section 8.4. Section 8.5 is the conclusion.

8.1 Introduction

There are two further questions which must be asked to answer the question stated at the start of the chapter. The first question is "what role does the Ant algorithm play when combined with a local search method?" The second question, which follows

on from this is "can the Ant algorithm be modified to improve performance?" This chapter will attempt to answer the first, and suggest some possible modifications that could achieve the second.

In Chapter 5 three Ant algorithms were used to solve various problems without the aid of a local search method. This was achieved because they were assuming the following:

Assumption: Better solutions can be found near known good solutions.

This assumption was the basis for the pheromone matrix which is a memory of good relationships between solutions. Thus the construction method for solutions is trying to build better solutions from parts of old ones. In general, this produces solutions that are similar to the global best but not identical, although these solutions are sometimes better than the original.

When a local search method is added it is normally included after the construction function and before the pheromone matrix update function of the algorithm. Therefore there are two roles the Ant algorithm can play:

- either to find solutions with many good components but which require a small adjustment to achieve a better objective value, or
- to find solutions that will enable the local search method to find a better global best

At the moment the field believes the first to be the case and that the local search method does not affect this role. This chapter will give evidence that, by assuming this, the Ant algorithm does not make full use of the local search method it is combined with. Therefore changes should be made to convert the Ant algorithm's role to the latter.

Related to this idea is the way in which researchers have tried to show that the pheromone matrix helps the local search produce better solutions than random search. In previous studies the following 3-step logic has been used:

1. *Experiment with Ant algorithm X combined with local search on problem P, produce best known result.*
2. *Experiment by turning off the pheromone matrix (setting $\alpha = 0$) in Ant algorithm X, which is combined with local search. Run again on problem P, and observe that it does not produce best known result.*
3. *Therefore, the Ant algorithm adds some quality Y that enables it to find new best*

known result.

The quality Y is concluded to be the new pheromone matrix or pheromone update rule used in the Ant algorithm X as this is what was removed. However, this does not take into account that when α is set to zero, not only is the pheromone matrix information lost but the link between the construction of new solutions and the global best solution is also broken. Therefore, it does not take into account that Ant algorithms might work because they generate solutions around the global best, so that it is acting like a variable neighbourhood search. This is clearly the case and the real question is whether the pheromone matrix offers the right kind of variation for driving the local search method. This chapter shows that Ant algorithms do not produce the kind of solutions that local search requires to find new local optima, and that the two roles mentioned above are in conflict.

8.2 A Shaking Algorithm

To evaluate the quality of the neighbourhood of solutions produced by an Ant algorithm it is necessary to have a comparison. For this a *Shaking* algorithm is implemented that can perform a fixed random neighbourhood search and a simple variable random neighbourhood search. The term *shaking* is used to emphasise the idea that the solutions generated around the global best are not constructed in the same way as in Ant algorithms, but by randomly moving components around in the global best solution.

To evaluate the variation in the solutions produced by each algorithm the distance of each solution will be calculated from the global best via the *Binary Manhattan Distance*. This distance is the number of positions where the solutions do not have the same component. For the TSP the solutions are synchronised at the node 0 and then the Manhattan distance is calculated, see Figure 8.1.

The shaking algorithm that is implemented here will be referred to as the Shaking Search (SS). It is used to test whether the pheromone matrix provides an effective neighbourhood search. Effective in this context refers to the desire that the Ant algorithms should be able to find a distribution of solutions that is better than the simple fixed and variable shaking searches. The algorithm is detailed in Figure 8.2.

To construct new solutions the Shaking Search moves components around in a solution. A *move* is simply a rearrangement whereby component *i* is picked at random (in this

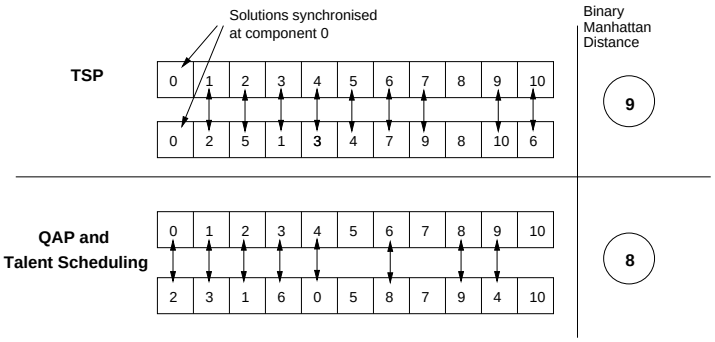


Figure 8.1: Illustration demonstrating the Binary Manhattan Distance used to calculate the difference of a new solution from the best solution found at a certain point in time.

case a uniform distribution is used), and then moved to a destination location j, also picked at random with the same distribution. The components in locations less than or equal to j are moved up or down as necessary to fit component i in. The solution is then repaired if necessary, for instance in the TSP one must ensure that the last component is equal to the first to create a round trip. This process of performing a number of *moves* is termed *shaking*, as the solution is shaken about to dislodge it from a local optimum.

In the original algorithm that was tested the components in the global best solution were *swapped* instead of *moved*. This was not as productive for the algorithm because some solutions that would be close for the Ant algorithm to reach could not be reached within a small percentage of n generations. Thus moves were selected so that all the solutions created by the Ant algorithm's construction method could be reached by the *shaking* method.

All the algorithms were run for a set number of iterations. However, the *end conditions* are the same as for the Ant algorithms. These are:

- when the specified number of iterations has been reached,
- when the specified amount of time has been run for,
- the optimum is reached,
- or the algorithm stagnates for some period of time.

Shaking Search has two advantages over the Ant algorithms, both are a consequence of dropping the pheromone matrix. The first advantage is speed of constructing new solutions. It is clear that moving a piece from location i to a different location j a

Figure 8.2: Shaking Search (SAS)

```
1  S ← ∅ // set of solution, objective value pairs ;
2  s_gb ← ∅;
3  v_gb ← 0;
4  n ← number of components per solution;
5  while end conditions not reached do
6      if t = 0 then
7          for (k = 1;k <= m; k++) do
8              s_k ← constructSolution();
9              S ← {s_k, f(s_k)};
10         endfor
11     else
12         for (k = 1;k <= m; k++) do
13             s_k ← s_gb;
14             w ← k mod (n+1);
15             for (i = 0;i < w; i++) do
16                 src ← random(n);
17                 dest ← random(n);
18                 move(src,dest);
19             endfor
20             S ← {s_k, f(s_k)};
21         endfor
22     endif
23     (s_gb, v_gb) ← min(S);
24 endw
25 return s_gb
```

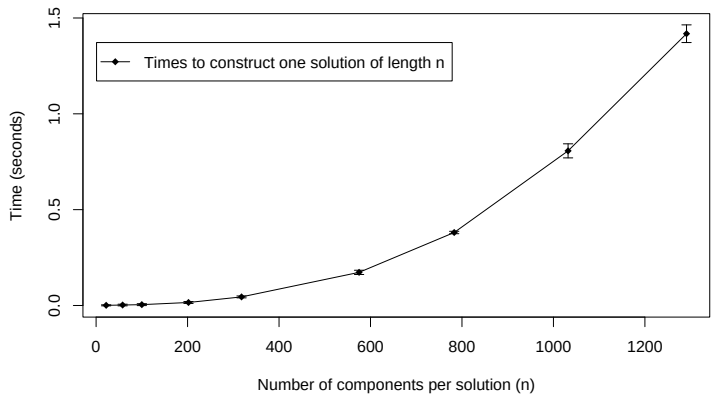


Figure 8.3: Graph showing how the time to construct one solution, using the Ant algorithm construction method, rapidly increases with the number of components (for example cities in the TSP) when using no nearest neighbourhood parameter to limit the branching factor.

fixed number of times can be done more quickly than constructing a new solution from scratch (the growth of which is illustrated in Figure 8.3). The algorithm keeps the graph construction method of the Ant algorithms to generate a initial set of solutions, S_0. However, because the pheromone matrix is uniform at this stage it is a method of producing m random solutions. After the first iteration, the solutions are generated via a fixed number of moves directly from the current global best solution found by the algorithm.

For the variable neighbourhood search the algorithm produces m solutions, each a different number of moves away from the global best. For instance, if $m = 10$ the solutions would range from 1 to 10 moves away from the global best. If m is set to more than n then the modulo is taken so that more ants will search in close proximity to the global best solution at first but then further out as m is increased. This is depicted in Figure 8.4. The consequence of this can be seen in the comparison to the other three algorithms in Figure 8.5. In this figure the mean Binary Manhattan distance from the global best is shown with the standard deviation depicted by the error bars.

For the fixed neighbourhood search the number of moves is fixed for all ants using a parameter n_{shake}. Therefore if $m = 10$ all ants produce solutions n_{shake} moves away from the global best. This parameter can be in the range $[1, N]$, where N is the size of the problem.

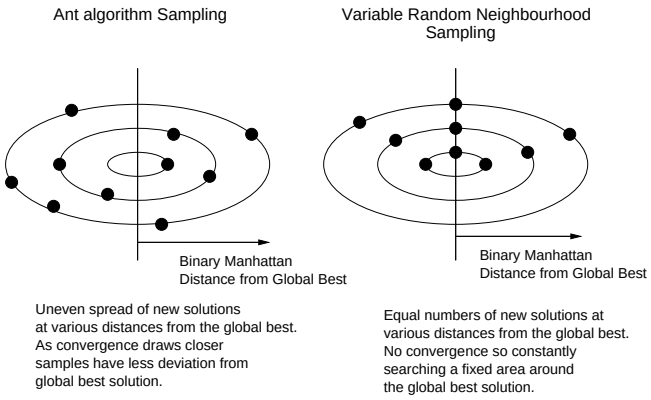


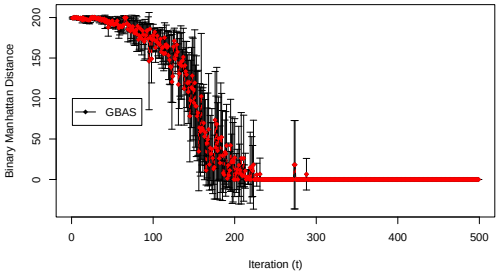
Figure 8.4: Illustration showing, for each style of algorithm, the sampling around the global best is carried out.

The second advantage of using the Shaking Search is that less memory is used. This saving is made because no pheromone matrix is required. This normally uses an amount of memory equal to N^2, which for very large problems can be a significant consideration. The amount of memory used in Shaking Search is exactly the same length as the best solution found so far, which is of length N.

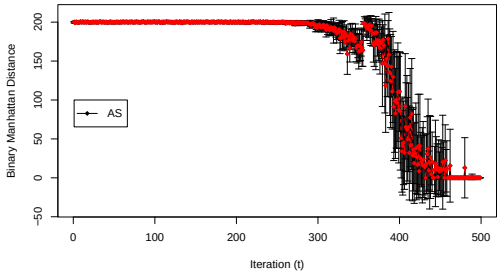
Figure 8.5 shows the variation in solutions generated by $m = 10$ ants using each different algorithm's construction method. These distances are recorded before the local search method is used on any of the solutions. Figure 8.5(a) shows a typical run for the Graph-Based Ant System. It can be seen that at the beginning the solutions are at their most diverse, then over time the diversity is expanded and then the algorithm focuses on the global best, until at about $t = 220$ the algorithm has converged and is no longer able to generate new solutions.

The graph for Ant System, shown in Figure 8.5(b), has a longer period of time at the most diverse level, almost 300 iterations. It is only then that the slope occurs, as seen in the GBAS figure. In Figure 8.5(c) the graph for MMAS is depicted and shows a different pattern to the other two algorithms. Here, again, there is a period of diversification at the start, of a duration between that of GBAS and AS, but then as the mean distance decreases the standard deviation increases so that different solutions up to a Binary Manhattan distance of 200 are still being generated. This difference would account for the increased performance of MMAS compared to GBAS and AS in Chapters 5 and 7.

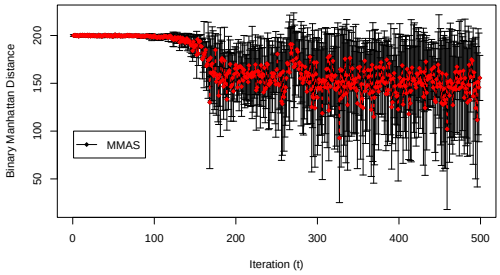
The final graph shown in Figure 8.5(d) is the Shaking Search with the variable neigh-



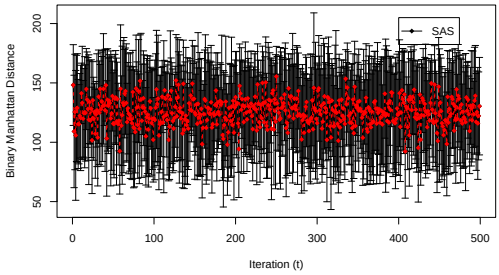
(a) GBAS



(b) AS



(c) MMAS



(d) Variable Neighbourhood SS

Figure 8.5: Graphs illustrating the sampling of solutions around the best solution found at time t on the TSP problem gr202.

bourhood. The mean distance is kept steady at approximately 125, with a standard deviation that covers from a distance of 50 to 200. This is approximately the same as the Max-Min Ant System but it is constant and does not diminish. This is the reason why, when mixed with a local search method, Shaking Search provides a significant simplification from the Ant algorithms. It also explains why, when not mixed with local search, Shaking Search is not useful at all. Without the local search method the Shaking algorithm cannot make small enough changes to find local optima.

Ant algorithms can be extended using various techniques, such as modifying the distribution of where components are moved to by using a heuristic, or cutting the number of possible moves by using the nearest neighbour parameter, or by using random restarts. It would also be possible to attach all these extra techniques on to any other algorithm, but they too have been kept simple to allow the focus to be on the differences.

	Algorithms		
Problem	GBAS	AS	MMAS
TSP	$\alpha = 1, \rho = 0.05$	$\alpha = 1, \rho = 0.95$	$\alpha = 1, \rho = 0.95$
QAP	$\alpha = 1, \rho = 0.05$	$\alpha = 2, \rho = 0.9$	$\alpha = 1, \rho = 0.9$
TS	$\alpha = 1, \rho = 0.05$	$\alpha = 2, \rho = 0.9$	$\alpha = 1, \rho = 0.7$

Table 8.1: Parameters to be used for the Ant algorithms based on Chapter 7.

8.3 Experimental Methodology

To perform a fair comparison of Shaking Search with the three Ant algorithms the best parameters from Chapter 7 were used. These values can be seen in Table 8.1. No additional techniques were used so that the Ant algorithms were not biased in favour of better solutions, this included not using heuristics, or parameters such as q_0 or n_0 to introduce diversity into the algorithms.

Initially the number of iterations used for the experiments was 500, the same number as in the three previous chapters. This was later increased so that the Ant algorithms would be given the best possible chance to find their best solutions.

In total, 36 problems were chosen with a wide range of sizes. The number of problems in a domain was varied to get the best impression of how the algorithms coped with each domain. For Talent Scheduling it was found that the Ant algorithms performed well with the problems and therefore the emphasis was placed on the Shaking Search to get similar results. In contrast, for the QAP results varied and therefore more problems were used to get a definite decision on whether Shaking Search could perform as well as the Ant algorithms.

- 9 TSP problems, problem chosen at random from medium sized TSP group ($100 < n < 1000$).

  lin105, gr120, ch150, gr229, lin318, gil262, pr264, rd400, rat783

- 22 QAP problems, problems chosen at random from each QAP group.

  UIGD: rou15, nug28, sko100b, sko100d
  URGI: chr25a, tai40a, lipa60a, lipa90b, tai100a, wil100
  RLI: esc16c, bur26d, bur26h, kra30b, esc32f, ste36b, esc64a, esc128
  RLLI: tai50b, tai60b, tai80b, tai100b

- 5 TS problems, one problem chosen at random from each size category {70, 100, 150, 175, 200}.

  talent_10_70_8, talent_10_100_19, talent_10_150_10, talent_10_175_1, talent_10_200_18

Each problem was run 25 times with each algorithm, thus 3600 runs were performed in total. The distribution of results were then tested for significance using the t-test. This test was used because there were many equal results, which made non-parametric tests ineffective. The hypotheses being tested were based on the following template:

Null Hypothesis:	There is no significant difference between the objective values of the best solutions found by the Shaking Search and GBAS when run on TSP problems.
Alternative Hypothesis:	The distribution of objective values is greater for the Shaking Search than for GBAS when both are run on TSP problems.

For both other hypotheses the TSP can be swapped for QAP and Talent Scheduling, while the algorithm GBAS can be swapped with AS and MMAS. Therefore this chapter tests a total of nine hypotheses.

In this chapter the only concern is the best objective value found by the algorithm. This is because the aim is to refute the claim that Ant algorithms can achieve *better* solutions than these simpler methods. The experiments compare the distributions of the best objective values of each algorithm rather than the best of the best objective values. This is because all of the algorithms are subject to random fluctuations and therefore any of them may stray upon a very good solution once within the 25 trials but to consistently do this would be significantly improbable. All the runs are performed with $m = 10$ ants.

8.4 Results

The first set of problems were run with the following parameter settings:

- $max_{it} = 500$

In Table 8.2 the results of the t-tests are given and it shows that the algorithms were split over performance. The TSP problems were not solved significantly better by the

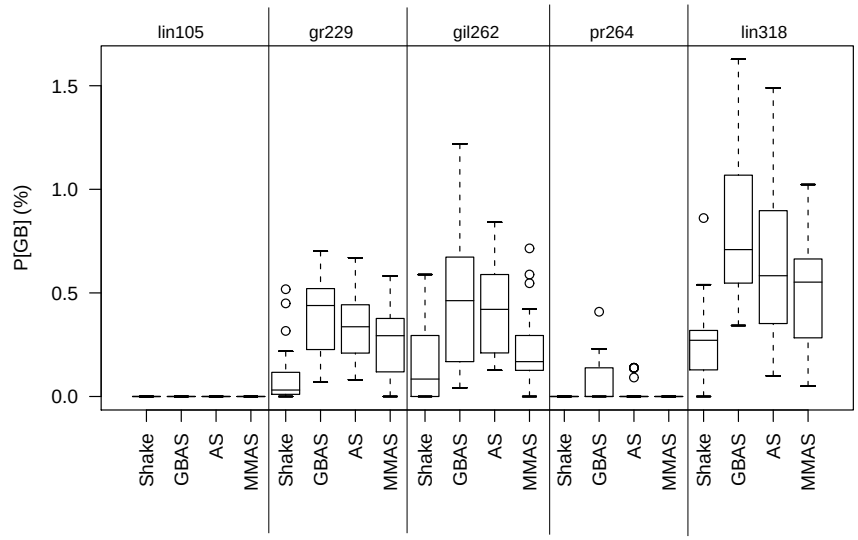


Figure 8.6: Boxplot showing the performance of the Shaking and Ant algorithms on the TSP problems.

Ant algorithms than the Shaking Search, with 100% of the distributions failing to show a significant difference. However, Figure 8.6 shows that the Shaking Search outperformed all the Ant algorithms, and not only this but it did so with smaller variances.

For the QAP set of problems the results were more mixed. Only for the Real-Life Like Instances (RLLI) did the Ant algorithms find significantly better solutions than the Shaking Search. For these two problems the variances were much wider for the Shaking Search, although it can be seen from Figure 8.7 that this did not mean it could not find solutions of the same quality as the Ant algorithms.

The final set of problems tested were the Talent Scheduling problems. Here the result was a definite success for the Ant algorithms. Only a few results indicated that the Shaking Search could match the other three. In Figure 8.8 shows that the Shaking Search's results have a much wider distribution and that in some cases the Ant algorithms managed to find solutions the Shaking Search could not.

Following the results of the previous experiment the number of iterations was increased from 500 to 2000. This was to try and ascertain whether either algorithm type was being hindered by too few iterations. Another reason for performing more runs was to increase the size of the TSP problems and to test more QAP problems to see if the size or quality of problem was the reason for the previous results. All of the other conditions remained the same. Table 8.3 displays the t-test results for these problems.

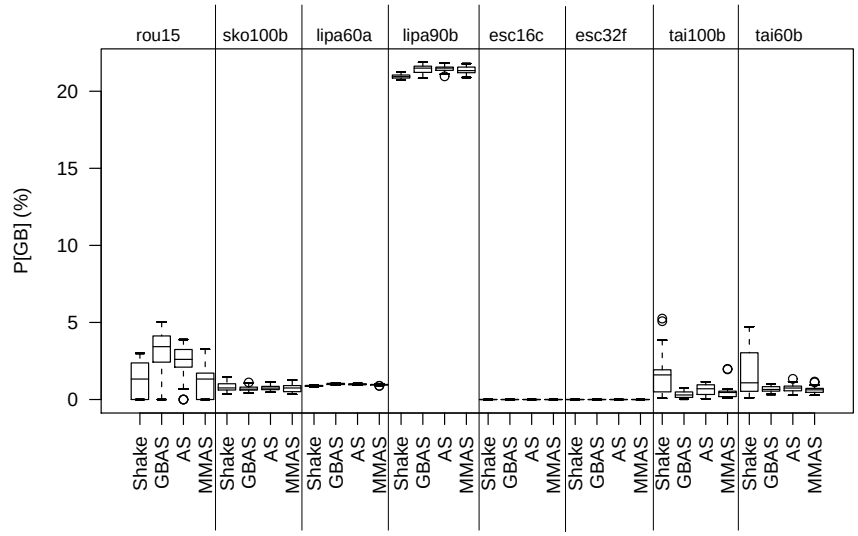


Figure 8.7: Boxplot showing the performance of the Shaking and Ant algorithms on the QAP problems.

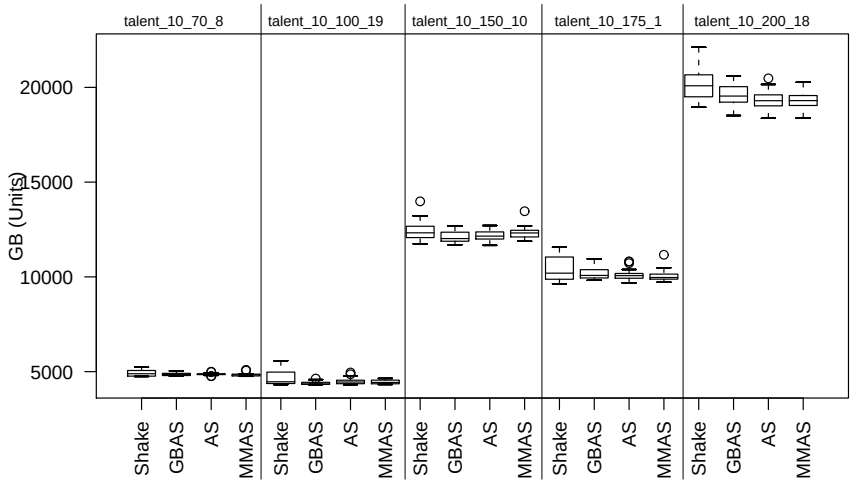


Figure 8.8: Boxplot showing the performance of the Shaking and Ant algorithms on the Talent Scheduling problems.

Group	Problem	GBAS	AS	MMAS
	lin105	1.0000	1.0000	1.0000
	gr229	1.0000	1.0000	0.9998
	gil262	1.0000	1.0000	0.9999
	pr264	0.9974	0.9940	1.0000
	lin318	1.0000	1.0000	0.9997
UIGD	rou15	1.0000	0.9993	0.4351
UIGD	sko100b	0.0829	0.2374	0.1939
URGI	lipa60a	1.0000	1.0000	1.0000
URGI	lipa90b	1.0000	1.0000	1.0000
RLI	esc16c	1.0000	1.0000	1.0000
RLI	esc32f	1.0000	1.0000	1.0000
RLLI	tai100b	0.0005*	0.0007*	0.0003*
RLLI	tai60b	< 0.0001*	0.0012*	0.0004*
	talent_10_70_8	0.0400^{*B}	0.0623	0.0175^{*B}
	talent_10_100_19	0.0014*	0.0328^{*B}	0.0067*
	talent_10_150_10	0.0022*	0.0135^{*B}	0.1915
	talent_10_175_1	0.0680	0.0207^{*B}	0.0061*
	talent_10_200_18	0.0014*	< 0.0001*	< 0.0001*

Table 8.2: Table showing the p-values to 4 decimal places of the outcome of the t-test calculations testing whether the Shaking Search solutions were worse than the Ant algorithm solutions. Runs lasted for a maximum of 500 iterations. (*=Significant to 95% confidence interval. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is as follows: TSP, TS is $\frac{0.05}{5} = 0.0100$, and QAP is $\frac{0.05}{8} = 0.0063$.)

In the case of the TSP problems, as for the previous experiment, all the t-tests indicated that the variable neighbourhood search was easily matching the quality of the Ant algorithms. For the QAP, fourteen different problems were tested, and the results were again mixed. The Ant algorithms only performed better on the Real-Life and Real-Life Like problems (RLI and RLLI sets). For the sets UIGD and URDGI, these random problems showed that the Shaking Search performed as well as the Ant algorithms. Unlike in the last experiment it is worth observing that not all the Ant algorithms, for a particular problem, are significantly better. In the case of ste36b only MMAS can produce a better distribution than the Shaking Search. This is of interest as it may be that some Ant algorithms are better suited to certain problem landscapes than others.

Finally, the Talent Scheduling problem results were more mixed than in the previous tests. This shows that with more iterations the Shaking Search was able to better compete with the Ant algorithms, but the general outcome was the same. For these problems the Shaking Search was out-performed by the Ant algorithms.

It is clear that the variable neighbourhood that was chosen for the Shaking Search is not suited to Talent Scheduling nor the QAP sets, RLI and RLLI. The Ant algorithms, although restricted to 10 ants, were still able to create wider variation in their solutions. Therefore it seemed appropriate to investigate whether a fixed neighbourhood could be found for each problem where Ant algorithms performed well and which would enable the Shaking Search to compete.

Hence line 14 of Figure 8.2 was changed so that a fixed integer was assigned to w. This value was varied from 1 to n to find the smallest number of moves needed to compete with the Ant algorithms. Each run was allowed to continue for a maximum of 2000 iterations and 25 runs per problem were computed. These were then compared using a t-test to see if the Shaking Search performed any worse than the Ant algorithms.

The results of this are given in Table 8.4. The fact that a fixed neighbourhood search can perform as well as Ant algorithms shows that the problem landscapes are not very complicated. This is the simplest neighbourhood search, yet it can perform as well as a more complex algorithm in these two domains. For each problem a particular number of shakes w was found that could produce solutions of the same quality as the Ant algorithms, except for tai60b for which no w could be found that could beat MMAS.

For these 12 problems the following data was collected.

- Let IB_t be best solution constructed by an algorithm at cycle t before the local

Group	Problem	GBAS	AS	MMAS
	gr120	1.0000	0.9999	1.0000
	ch150	1.0000	1.0000	1.0000
	rd400	1.0000	1.0000	1.0000
	rat783	1.0000	1.0000	1.0000
UIGD	nug28	1.0000	1.0000	0.9718
UIGD	sko100d	0.2416	0.9705	0.1672
UIGD	wil100	0.4485	0.9627	0.1495
URGI	chr25a	1.0000	1.0000	0.9998
URGI	tai40a	1.0000	1.0000	1.0000
URGI	tai100a	1.0000	1.0000	1.0000
RLI	bur26d	0.0003*	0.0004*	0.0001*
RLI	bur26h	0.2438	0.2330	0.1622
RLI	kra30b	0.9659	0.9995	0.0032^{*B}
RLI	ste36b	0.8098	1.0000	0.0138^{*B}
RLI	esc64a	1.0000	1.0000	1.0000
RLI	esc128	1.0000	0.9786	1.0000
RLLI	tai50b	0.0004*	0.0063^{*B}	0.0002*
RLLI	tai60b	0.0080^{*B}	0.0172^{*B}	0.0004*
RLLI	tai80b	0.0154^{*B}	0.2436	0.0005*
RLLI	tai100b	0.0222^{*B}	0.1120^{*B}	0.0124^{*B}
	talent_10_70_8	0.0762	0.1514	0.0049*
	talent_10_100_19	0.0110^{*B}	0.0520	0.0036*
	talent_10_150_10	0.0396^{*B}	0.3087	0.7292
	talent_10_175_1	0.0737	0.2399	0.0369^{*B}
	talent_10_200_18	0.0001*	0.0027*	0.0986

Table 8.3: Table showing the p-values to 4 decimal places of the outcome of the t-test calculations testing whether the Shaking Search solutions were worse than the Ant algorithm solutions. Runs lasted for a maximum of 2000 iterations. (*=Significant to 95% confidence interval. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is as follows: TSP is $\frac{0.05}{4} = 0.0130$, TS is $\frac{0.05}{5} = 0.0100$, and QAP is $\frac{0.05}{16} = 0.0031$.)

Problem	Num. Of Shakes
bur26d	6
kra30b	4
ste36b	7
tai50b	14
tai60b	59*
tai80b	20
tai100b	8
talent_10_70 _8	10
talent_10_100_19	10
talent_10_150_10	40
talent_10_175_1	7
talent_10_200_18	30

Table 8.4: Table showing the minimum number of shakes required for a fixed-neighbourhood version of the Shaking Search to find solutions as good as the Ant algorithms' solutions. Runs lasted for a maximum of 2000 iterations. (*=indicates that it was not as good as MMAS)

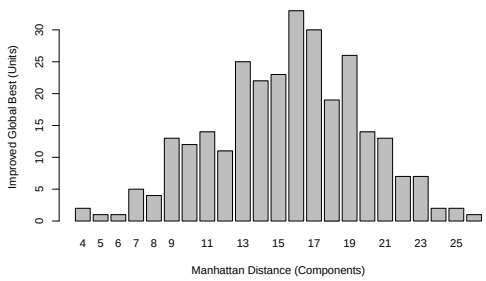
search method is used.

- Let $\delta(IB_t)$ be the Manhattan distance of this solution from the global best solution.
- Let $L(\cdot)$ be the local search method that takes a solution as its argument.
- Let $I(\cdot)$ be a function that returns true only if $L(IB_t)$ is better than the global best solution.

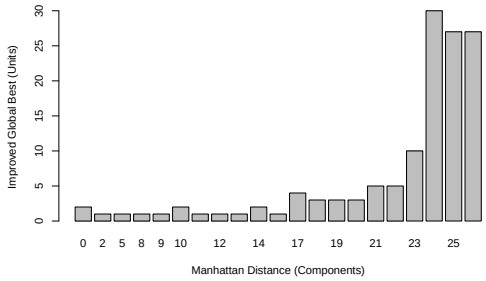
By collecting the triple $(t, IB_t, \delta(IB_t))$ when $I(L(IB_t))$ was true, graphs could be constructed to show what kind of solutions produced improvements for each algorithm. To illustrate the results bur26d was chosen.

Figure 8.9 shows how many solutions led to improved global bests against the Manhattan distance of the seed solution to the local search from the previous best solution. It is clear that both GBAS and AS have little representation in the majority of the lower to middle ranges. MMAS, which is consistently the closest competitor to the Shaking Search, does produce solutions in a broader range.

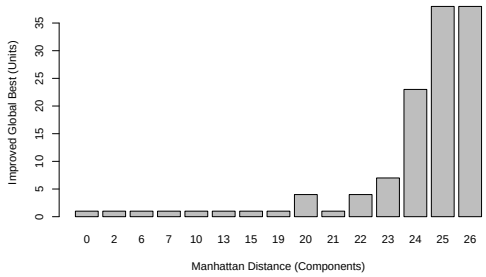
Figure 8.10 shows the same problem with $L(IB_t)$ plotted against $\delta(IB_t)$. These graphs show when particular distances were successful at allowing the local search to find new local optima for a seed of a certain quality. This shows that improvements, even when the solution is close to the global optimum, occur at all distances from the current best.



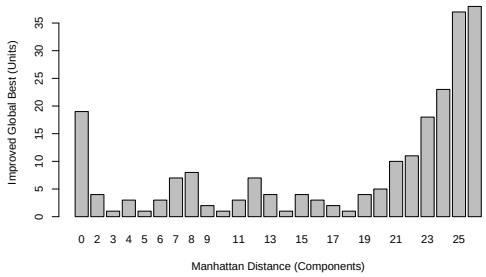
(a) Fixed-Neighbourhood Shake



(b) GBAS



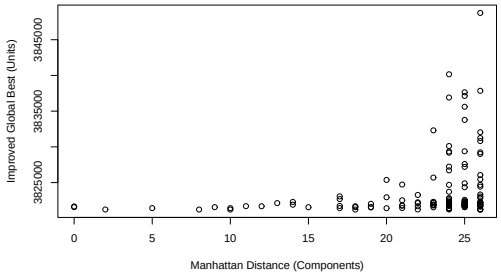
(c) AS



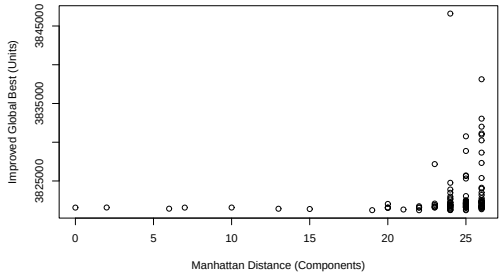
(d) MMAS

Figure 8.9: Figure showing how many solutions led to improved global bests against the Manhattan distance of the seed solution to the local search from the previous best solution, results for the TSP problem `bur26d`.

(a) Fixed-Neighbourhood Shake



(b) GBAS



(c) AS

(d) MMAS

Figure 8.10: Figures illustrating that better solutions are not necessarily found near the global best solution when using a local search method.

Both Figures 8.9 and 8.10 show that the Ant algorithms are making the following assumption:

Assumption: Better solutions lie close to the Global Best.

Thus they are acting as solution optimisers. The problem with this assumption is that it is only valid, *if the best solution per iteration is based on the best solution from the previous iteration.* When a local search method is introduced it breaks this link between the cycles. Therefore the assumption being implemented when a local search method is introduced is:

Assumption: Better solutions for the local search method to act upon lie close to the Global Best.

This assumption is not valid and it is this assumption that means that Ant algorithms are not as good at driving local search methods as they could be. Referring back to the roles discussed in the Section 8.1, it is clear that there is a conflict of interests within current Ant algorithms when they are combined with a local search method. From these results it is clear that if Ant algorithms were to concentrate on the second role and learn how to generate solutions which the local search can then optimise this would lead to better results.

8.5 Conclusion

This chapter attempts to explain how Ant algorithms work with local search methods to produce good quality solutions.

The benefits of creating solutions from a construction graph is that it allows one to use the pheromone matrix costs allocated to each arc and to use heuristics in an easy manner. When local search is used these two advantages have to be defended. The former is defendable as long as one can show that the pheromone matrix adds some contribution to the search effort. On the other hand, it has been shown in Chapter 7 that the heuristic does not make a significant difference to the solution quality when a local search method is used.

In this chapter an algorithm was implemented that kept the global best solution and generated new solutions in a neighbourhood about this solution, but without the aid of

the pheromone matrix. This algorithm was then shown to be as effective as any of the Ant algorithms on TSP problems.

The algorithm was then modified to a fixed neighbourhood search to solve the QAP and Talent Scheduling problems to the same quality as the Ant algorithms. This gave evidence that showed that simpler algorithms could be just as effective as an Ant algorithm.

The primary conclusion from this chapter is that Ant algorithms are not good local search drivers because they do not allow for the fact that, although good solutions may be found near the global best, these same solutions are not varied enough to offer the local search method new areas of the space to optimise.

This opens the way for better algorithms in the future based on ideas from both Ant algorithms *and* simple Shaking algorithms around a single solution memory. Ant algorithms offer reliability of solution quality, Shaking algorithms offer better solution quality and speed of construction. The objective of such an algorithm would be to learn which solutions are good for generating better solutions via the local search rather than trying to act as a solution optimiser.

8.6 Summary

This penultimate chapter of the thesis shows that Ant algorithms must be altered to take full advantage of the local search methods they are often combined with. It has shown evidence that the assumptions which allowed Ant algorithms to work alone are not always valid when acting as part of a hybrid algorithm. The following chapter concludes the thesis.

Chapter 9

Conclusions

Now that all the results have been discussed, in this chapter the contents of the thesis will be summarised to emphasise the main conclusions. Following this in Section 9.2 there will be a discussion of how this work impacts on the community. Finally, an examination of how this work could be taken forward in the future will be presented.

9.1 Thesis Synopsis

The question this thesis set out to answer was "how is knowledge used by Ant algorithms?" To answer this question it was necessary to begin by reviewing all the Ant algorithm work that has been published to see what trends had developed and what the most influential algorithms had been. Chapter 2 revealed that many algorithms had been developed using the Ant metaphor, taking influences from Reinforcement Learning, Branch-and-Bound and Genetic Algorithms. Many papers showed that their particular algorithm outperformed Ant System, Max-Min Ant System or other different algorithms. Therefore it was decided to find an algorithm that could represent all these and make the work in this thesis applicable to as much of the field as possible.

The Graph-based Ant System (GBAS), by Walter Gutjahr, was a theoretical model of what an Ant System should be like. He had shown that his model would converge to the optimal solution with some probability close to 1 given enough time. Ant System had been assumed to work in the same way as this model, while Max-Min Ant System had been shown to work slightly differently and therefore the convergence proof could not be applied. However, for both algorithms there were great similarities and so the

first step was to implement GBAS and to show that under certain conditions both Ant System and Max-Min Ant System are modelled successfully. These two algorithms were chosen because they have been the most influential in the field.

For all the experiments three domains had been chosen: Travelling Salesman Problem (TSP), Quadratic Assignment Problem (QAP) and Talent Scheduling (TS). The first two were chosen for their popularity from other research, while the third was chosen to provide variety. The exploratory experiments were performed using TSP only, and then random problems were selected from the QAP and TS problem sets to remove any bias that might have been introduced in the problem selection.

The first objective of the thesis was to answer the question "Is the Graph-based Ant System representative of both Ant System and the Max-Min Ant System?" Thus in Chapter 5 Ant System and Max-Min Ant System were studied to see how well they were modelled by the GBAS. It was found that generally both algorithms could be assumed to behave like GBAS.

The primary exception was when ρ_{alg}[1] > 0.9 which led to the models behaving very differently. On the one hand GBAS's performance would continue to improve, while the other two algorithms would degrade. It was found that the optimal values for α and β for all three algorithms was 1 and 2 respectively, for domains where a heuristic was available. It was also discovered that performance was very similar on the TSP domain, but for QAP and TS performance was algorithm dependent. For all the algorithms it was shown that low number of ants were the ideal setting, implying that regular updates are much more useful than large samples. This characteristic is likely to be linked to the problems and could change with more varied landscapes.

Overall the chapter showed that the algorithms were similar, except when the parameters were moved to extremes. This effectively showed that the pheromone matrix configuration, the internal knowledge source, was critical only in absolute performance terms and at parameter extremes. In terms of how the algorithms reacted to parameter variation there was a high degree of similarity. Thus the answer to the first objective was that GBAS is a good model for how both Ant System and Max-Min Ant System work.

The thesis then moved on to the first external knowledge source which was the heuristic. This chapter asked the question "What effect do misleading heuristics have on

[1] ρ_{alg} is a substitution for ρ in Ant System and MMAS, and $1 - \rho$ in GBAS.

the three Ant algorithms?" Therefore, Chapter 6 described how five random heuristics could affect the performance and convergence characteristics of the Ant algorithms. Again, all three algorithms were used as it was unclear whether using misleading heuristics would make the algorithms behave differently. In general this was not the case, thus the answer to the question above was that all the algorithms perform badly with misleading heuristics.

The experiments showed that the algorithms could not learn to tell good guidance from bad. However, using dynamic heuristics that could react to the state of the pheromone matrix could allow the algorithms to explore as well as exploit the landscape. At the end of the chapter a discussion gave some possible alterations to the algorithms that would allow for differentiation between the quality of the heuristic guidance.

Before the experiments in the next chapter could be performed it was necessary to write a local search method for the Talent Scheduling problem. Therefore in Chapter 4 an incremental local search method was described that would allow small changes in the solution to be evaluated in an efficient manner. It was found that there was little difference between the quality of solutions found with the First-Improvement and Best-improvement methods, thus the First-Improvement method was chosen as it was the fastest.

In Chapter 7 the second external knowledge source was investigated. It asked the question "Does local search alter the properties of the three Ant algorithms?". Local search methods have been combined on numerous occasions with Ant algorithms and yet no study had been made to see if this was a good fusion. The outcome of the chapter was that local search does significantly alter the properties of the three Ant algorithms from those found in Chapter 5 in a variety of ways.

For instance, in many of the problems attempted, only a single ant was required to find the best solutions, and in some cases a value of $\rho_{alg} = 0.2$ could find adequate solutions. When local search was added the algorithms became much more sensitive to the individual problem and the quality of the final solution was almost independent of any heuristic information good or bad that was given to the Ant algorithm.

Therefore, in Chapter 8 the relationship between Ant algorithms and local search was examined. This chapter asked "What is the reason for any incompatibility between Ant algorithms and local search?" To answer this question a Shaking algorithm was developed which constructed solutions from the global best solution, via a random

number of component moves. It was shown that even with the best set of parameters from Chapter 7 this much simpler algorithm could find solutions equally good to those from the Ant algorithms. The chapter demonstrated that the assumption made by all Ant algorithms that better solutions can be found near the current global best solution was not applicable when local search was being used. What was required from the Ant algorithm was not solutions composed of many good components that could be shaped into a better solution but, solutions that the local search method could optimise to a new local optimum. A great deal of time was being wasted running local search on similar solutions with the only result being that the local search would then move to the same local optimum as before.

In conclusion, the four objectives set out in Section 1.3 have been answered, with the following important conclusions for the community.

- As long as the parameters used are not at the extremes of the ranges allowed, most Ant algorithms based on Ant System and Max-Min Ant System changes to parameters will produce similar changes in performance.
- GBAS is a good theoretical model for how an Ant algorithm should work and is therefore worth pursuing for further theoretical work.
- Ant algorithms can only be used with heuristics that are known not to be misleading, otherwise the best course of action is to set β to 0.
- Local search does change the way Ant algorithms work and react to changes in their parameters.
- Ant algorithms and local search methods could be combined more effectively than they are currently.

9.2 Contributions to the Community

This thesis delivers a number of contributions to the field of Artificial Intelligence, especially to the area of Ant Algorithms. The topics below have been identified as future papers that would be of interest to the community.

9.2.1 Minor Contributions

- *A Local Search Implementation for the Talent Scheduling Problem*: This work gives the details of an implementation of a 2-Opt local search method for the Talent Scheduling Problem. It gives full details on the theory behind the local search and how it delivers an efficient algorithm. It then compares two policies, First-Improvement and Best-Improvement for solving a number of large random instances of this problem type. This is an important contribution as it will allow those developing other techniques to combine their implementation with the local search method, improving their results and allowing their techniques to scale more efficiently.
- *The First Implementation of the Graph-based Ant System*: This work introduces the first implementation of the theoretical Graph-based Ant System (GBAS) [Gutjahr, 1999]. In this work the implementation is described and performance-related results are given.
- *Graph-based Ant System as a Model for Ant System* and *Graph-based Ant System as a Model for Max-Min Ant System*: These two topics show in which circumstances the Graph-based Ant System is a good model for each of the influential Ant algorithm implementations. This work is important as it shows the link between the theoretical and the practical algorithms being developed in the field.
- *Heuristics as a Source of Error*: This work discusses how heuristics can be a source of error as well as guidance for Ant algorithms. It shows examples of heuristics feeding Ant algorithms misleading information and the impact on both convergence and performance. This is important for the community as heuristics are often picked with insufficient care, without the consequences being factored in.

9.2.2 Major Contributions

- *A Critical Analysis of the Field of Ant Algorithms*: This work is an overview of the way Ant algorithms have been developed over the last fifteen years. It analyses the content of previous papers in terms of their experimental method. As part of this analysis it examines the trends in the creation of new algorithms and highlights some areas of further study. This work is of benefit to the community as it relies

solely on the papers published and is written by a third party with no contact with the protagonists of the field. This fact enables the paper to be impartial and more critical than other previous overviews.

- *Heuristics as a Guide to Exploration*: This work introduces the idea that heuristics should be used not only for exploitation but for exploration as well. By using dynamic heuristics one can guide the Ant algorithm to more fruitful areas of the search space. This is an important topic as there is currently little study undertaken about using heuristics in this way, and this could help control the exploration more than other more arbitrary parameters.

- *The Consequences of Combining Local Search with Ant algorithms*: This work discusses the behavioural changes that occur when a local search method is combined with an Ant algorithm. This is of immense importance to the community as many papers report applications of such hybrid algorithms. It is important for the field to realise that there are changes to the parameter selections they might make and that these can effect the performance of their algorithms.

- *Understanding how Ant Algorithms work with Local Search*: This work discusses how Ant algorithms work with local search methods. It shows that Ant algorithms, without any external interference, are poor at driving local search. It demonstrates that a better understanding of the requirements of the local search method is required.

9.3 Further Research

This thesis has investigated some critical issues about the way Ant algorithms deals with knowledge from external sources. This work can be continued in a number of ways.

- *Direct extension of experimental efforts* - All the results in this thesis could be confirmed further with more domains and further Ant algorithms, such as the Population-based algorithms. This work could involve in each case investigating more combinations of parameters to extend the claims in this thesis.

  As this thesis has highlighted the selection of problems should be the central focus of future work. Much more difficult and more directed problems should be

attempted to identify where Ant algorithms are most useful.

- *Theoretical explanation of empirical phenomena*

  A number of decisions were made, such as to split ρ_{alg} into two groups $(0, 0.9]$ and $[0.9, 1.0)$ and also not to use n_0 and q_0. These decisions could be investigated further, and more importantly, the reasons for the two groups for ρ_{alg} could be studied in more detail. It would be of interest to study in more depth whether problem characteristics could affect this split. Furthermore, it would be interesting to know whether the behaviour in these two groups could be predicted a priori via theoretical means, especially if it was the result of the normalisation within the pheromone matrix.

  There is a great deal of empirical evidence in this thesis, regarding the interaction of local search and the parameters of Ant algorithms that may also benefit from a theoretical explanation.

- *The study of more responsive and context-sensitive heuristics*

  There are some interesting questions that still remain from Chapter 6. The emphasis of any continued work should be on developing more dynamic, and useful, heuristics that can be weighted according to their effectiveness.

  The ability to handle combinations of heuristics for a particular problem would also be of value. This value would come from the application of the appropriate heuristic given the state of the pheromone matrix. This would mean that the heuristic selection mechanism would have to be dynamic to react appropriately. Currently many heuristics are discarded for use with Ant algorithms as they do not give good guidance in the majority of situations. However, to be able to learn the appropriate situations in which to use these fragile heuristics would maximise their utility. This contextual learning is difficult and would require further study.

  At the end of Chapter 6 a number of strategies were discussed for weighting heuristics more effectively, and the implementation of these could form the basis of future investigations. The main problem is to weight the pheromone information and heuristic guidance on the same scale so that one does not come to dominate the other. It would also be desirable to have feedback that indicated when the heuristic chose a good component and when it did not, this might provide a better application context, as discussed in the previous paragraph.

- *The formal integration of Local Search with Ant algorithms*

  There is still a great deal not known about the interaction of Ant algorithms and local search and so there is a need for more research in this area. Designing new algorithms based on Ant algorithms that could harness the power of local search methods in a more controlled manner is an important area. This is especially important when the local search is optimising a relationship that the Ant algorithm is not able to represent within its pheromone matrix. The representation used by the Ant algorithm for the pheromone matrix should be chosen carefully with respect to the various manipulations that the local search method performs to optimise the benefit the Ant algorithm is awarded by the action of the local search method.

  There is a need for a more formal design process for putting the Ant algorithm, heuristics and local search methods together in such a way that each is optimised for the specific domain. This would greatly enhance the reputation of the field and enable application designers to apply these algorithms with confidence.

Appendix A

QAP Flow Dominance Values

Problem	Flow Dominance	Distance Dominance	Problem	Flow Dominance	Distance Dominance	Problem	Flow Dominance	Distance Dominance
bur26a	285.73	15.68	had18	49.90	67.43	sko100e	111.51	51.26
bur26b	285.73	16.54	had20	48.38	67.62	sko100f	109.43	51.26
bur26c	237.36	15.68	kra30a	155.07	50.89	sko42	111.10	53.21
bur26d	237.36	16.54	kra30b	155.07	51.69	sko49	111.64	52.61
bur26e	263.96	15.68	kra32	50.57	169.44	sko56	112.52	52.39
bur26f	263.96	16.54	lipa20a	47.99	41.83	sko64	110.08	51.99
bur26g	290.88	15.68	lipa20b	47.99	72.46	sko72	108.63	51.85
bur26h	290.88	16.54	lipa30a	44.39	33.27	sko81	107.93	51.57
chr12a	68.95	335.98	lipa30b	44.39	66.23	sko90	108.73	51.48
chr12b	68.95	335.98	lipa40a	43.15	28.45	ste36a	411.58	57.21
chr12c	68.95	335.98	lipa40b	43.15	65.78	ste36b	411.58	103.63
chr15a	74.72	350.31	lipa50a	42.00	25.25	ste36c	411.58	57.48
chr15b	74.72	350.31	lipa50b	42.00	63.01	tai100a	60.92	59.93
chr15c	74.72	350.31	lipa60a	42.81	22.93	tai100b	324.57	81.23
chr18a	66.81	371.22	lipa60b	42.81	61.47	tai10a	87.35	85.59
chr18b	60.21	377.28	lipa70a	42.52	21.15	tai10b	381.91	62.35
chr20a	62.51	364.15	lipa70b	42.52	61.22	tai12a	81.56	75.61
chr20b	62.51	364.15	lipa80a	42.79	19.73	tai12b	326.84	86.41
chr20c	69.08	364.15	lipa80b	42.79	60.92	tai150b	316.23	52.17
chr22a	70.07	440.65	lipa90a	42.31	18.57	tai15a	75.60	68.33
chr22b	70.07	440.65	lipa90b	42.31	60.70	tai15b	335.29	281.05
chr25a	60.34	441.59	nug12	127.18	62.06	tai17a	73.18	68.20
els19	559.74	54.92	nug14	111.53	61.11	tai20a	68.23	70.46
esc128	52.45	1163.03	nug15	114.08	60.62	tai20b	350.33	134.84
esc16a	90.27	181.39	nug16a	107.45	61.16	tai256c	218.75	260.69
esc16b	90.27	80.54	nug16b	123.30	58.42	tai25a	66.92	2536.90
esc16c	90.27	141.97	nug17	111.38	59.78	tai25b	323.08	90.57
esc16d	90.27	250.91	nug18	110.34	58.17	tai30a	65.35	59.97
esc16e	90.27	265.57	nug20	109.10	56.95	tai30b	334.89	88.09
esc16f	90.27	0	nug21	122.91	60.25	tai35a	63.36	63.43
esc16g	90.27	271.06	nug22	119.66	67.14	tai35b	318.60	80.94
esc16h	90.27	160.95	nug24	117.69	56.48	tai40a	61.76	64.70
esc16i	90.27	316.03	nug25	115.38	55.24	tai40b	325.25	68.44
esc16j	90.27	343.34	nug27	60.87	115.69	tai50a	63.51	61.98
esc32a	71.47	290.51	nug28	56.52	117.18	tai50b	320.26	74.93
esc32b	71.47	214.88	nug30	116.29	54.54	tai60a	61.88	62.44
esc32c	71.47	206.63	rou12	78.04	73.15	tai60b	323.17	78.12
esc32d	71.47	242.96	rou15	74.01	73.65	tai64c	129.85	489.70
esc32e	71.47	1126.06	rou20	67.74	69.02	tai80a	61.14	59.97
esc32f	71.47	1126.06	scr12	62.06	279.80	tai80b	327.24	64.85
esc32g	71.47	876.77	scr15	58.84	265.45	tho150	148.16	51.83
esc32h	71.47	193.82	scr20	56.95	267.72	tho30	142.54	61.26
esc64a	60.09	580.57	sko100a	107.71	51.26	tho40	159.48	54.55
had12	55.29	68.87	sko100b	109.46	51.26	wil100	65.16	51.26
had14	53.26	71.75	sko100c	109.16	51.26	wil50	68.01	55.29
had16	51.63	69.15	sko100d	110.29	51.26			

Table A.1: Table showing the Flow Dominance (fd) and Distance Dominance (dd) values for all problems found in QAPLIB

Appendix B

Tables for Chapter 5

	Problem	**AS,GBAS Correlation**		**MMAS,GBAS Correlation**	
ρ_{alg} **Range**		$(0,0.9]$	$[0.9,1)$	$(0,0.9]$	$[0.9,1)$
TSP					
	gr17	0.9639*	0.4589	0.7956^{*B}	$0.8991^{'B}$
	bayg29	0.9758*	0.4569	0.8229^{*B}	$0.8049^{'B}$
	brazil58	0.9385*	0.6511	0.7662^{*B}	$0.8450^{'B}$
	eil76	0.9486*	0.7211	0.7153^{*B}	$0.8424^{'B}$
	kroA100	0.9535*	$0.7761^{'B}$	0.7217^{*B}	$0.8733^{'B}$
QAP					
UIGD	had18	0.8943*	0.0979	0.5564	< 0.0001
UIGD	sko81	0.8869*	0.6820	0.7474^{*B}	0.6282
URGI	tai50a	0.9812*	0.3252	0.7768^{*B}	< 0.0001
URGI	lipa80a	0.9066*	0.2029	0.4899	0.1459
RLI	esc16c	0.8762*	< 0.0001	0.7577^{*B}	< 0.0001
RLI	bur26f	0.9071*	< 0.0001	0.7799^{*B}	< 0.0001
RLLI	tai15b	0.9935*	$0.7580^{'B}$	0.9243^{*}	< 0.0001
RLLI	tai100b	0.7554^{*B}	0.2138	0.8244^{*B}	0.4824
TS					
	talent_10_10_5	0.9241*	0.0848	0.7125^{*B}	0.2675
	talent_10_15_8	0.8697*	0.0054	0.7266^{*B}	0.3777
	talent_10_20_5	0.9320*	0.1854	0.7291^{*B}	0.4576
	talent_10_30_4	0.9664*	0.2445	0.7769^{*B}	0.5690
	talent_10_100_8	0.7566^{*B}	0.6728	0.6644	0.6905

Table B.1: Summary of results for the correlation for P_{GB} between GBAS and AS/MMAS varying parameter ρ_{alg}. Correlations calculated using the Pearson Product-Moment Correlation. (* = Significant correlation for $(0,0.9]$, degrees of freedom$= 9-2=7$, 0.05 significance$=0.669$, '= Significant Correlation for [0.9,1), degrees of freedom$=$ $6-2=4$, 0.05 significance$=0.729$, significance values taken from [Coolican, 1994]. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\alpha = \frac{0.05}{18} = 0.0028$.)

	Problem	**AS,GBAS Correlation**		**MMAS,GBAS Correlation**	
ρ_{alg} **Range**		$(0, 0.9]$	$[0.9, 1)$	$(0, 0.9]$	$[0.9, 1)$
TSP					
	gr17	0.9980*	0.3679	< 0.0001	0.2789
	bayg29	0.9990*	0.2840	0.4688	< 0.0001
	brazil58	0.9993*	< 0.0001	0.3752	< 0.0001
	eil76	0.9984*	< 0.0001	0.4188	< 0.0001
	kroA100	0.9997*	< 0.0001	0.4768	< 0.0001
QAP					
UIGD	had18	0.9742*	0.5651	0.3129	< 0.0001
UIGD	sko81	0.9977*	< 0.0001	0.2341	< 0.0001
URGI	tai50a	0.9923*	< 0.0001	0.7656	< 0.0001
URGI	lipa80a	0.9873*	< 0.0001	0.3583	< 0.0001
RLI	esc16c	0.9932*	0.6682	0.4854	0.0496
RLI	bur26f	0.9989*	< 0.0001	0.4276	< 0.0001
RLLI	tai15b	0.9905*	0.3942	0.4963	< 0.0001
RLLI	tai100b	0.9987*	< 0.0001	< 0.0001	< 0.0001
TS					
	talent_10_10_5	0.9939*	0.5702	0.3588	0.4620
	talent_10_15_8	0.9933*	0.5428	0.0268	< 0.0001
	talent_10_20_5	0.9961*	0.4601	0.5351	0.2817
	talent_10_30_4	0.9966*	0.1558	0.4534	< 0.0001
	talent_10_100_8	0.9983*	0.1271	0.3878	< 0.0001

Table B.2: Summary of results for the correlations for I_{GB} between AS/MMAS and GBAS varying parameter ρ_{alg}. Correlations calculated using the Pearson Product-Moment Correlation. (* = Significant correlation for $(0, 0.9]$, degrees of freedom$= 9 - 2 = 7$, 0.05 significance$=0.669$, '= Significant Correlation for [0.9,1), degrees of freedom$= 6 - 2 = 4$, 0.05 significance$=0.729$, significance values taken from [Coolican, 1994]. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\alpha = \frac{0.05}{18} = 0.0028$.)

Problem	β	r	P-Value	Problem	β	r	P-Value
gr17	0	0.76	0.0812	esc16c	0	0.68	0.1344
gr17	1	0.61	0.1983	bur26f	0	0.31	0.5508
gr17	2	0.66	0.1516	tai15b	0	0.73	0.1028
gr17	3	-0.22	0.6719	tai100b	0	-0.57	0.2346
gr17	4	-0.21	0.6879	talent_10_10_5	0	0.41	0.4178
gr17	5	NA (1)	NA(0)*	talent_10_10_5	1	0.29	0.5739
bayg29	0	0.82	0.0459^{*B}	talent_10_10_5	2	0.88	0.0223^{*B}
bayg29	1	0.98	0.0007*	talent_10_10_5	3	0.62	0.1914
bayg29	2	0.99	0.0003*	talent_10_10_5	4	NA (1)	NA(0)*
bayg29	3	0.99	0.0002*	talent_10_10_5	5	0.50	0.3129
bayg29	4	0.79	0.0618	talent_10_15_8	0	0.40	0.4279
bayg29	5	0.48	0.3307	talent_10_15_8	1	0.26	0.6182
brazil58	0	0.81	0.0491^{*B}	talent_10_15_8	2	0.29	0.5743
brazil58	1	0.98	0.0006*	talent_10_15_8	3	0.15	0.7739
brazil58	2	0.99	< 0.0001*	talent_10_15_8	4	0.04	0.9422
brazil58	3	0.99	0.0001*	talent_10_15_8	5	0.47	0.3430
brazil58	4	0.98	0.0004*	talent_10_20_5	0	0.71	0.1137
brazil58	5	0.93	0.0073^{*B}	talent_10_20_5	1	0.26	0.6165
eil76	0	0.88	0.0210^{*B}	talent_10_20_5	2	0.47	0.3451
eil76	1	0.97	0.0017*	talent_10_20_5	3	0.54	0.2654
eil76	2	0.99	0.0002*	talent_10_20_5	4	0.45	0.3659
eil76	3	0.99	< 0.0001*	talent_10_20_5	5	0.33	0.5289
eil76	4	0.99	< 0.0001*	talent_10_30_4	0	0.43	0.4006
eil76	5	0.99	< 0.0001*	talent_10_30_4	1	0.37	0.4748
kroA100	0	0.90	0.0155^{*B}	talent_10_30_4	2	0.46	0.3581
kroA100	1	0.97	0.0011*	talent_10_30_4	3	0.69	0.1259
kroA100	2	0.98	0.0004*	talent_10_30_4	4	0.71	0.1138
kroA100	3	0.99	0.0003*	talent_10_30_4	5	0.73	0.0996
kroA100	4	0.98	0.0007*	talent_10_100_8	0	0.48	0.3398
kroA100	5	0.99	< 0.0001*	talent_10_100_8	1	-0.42	0.4046
had18	0	0.43	0.3978	talent_10_100_8	2	-0.28	0.5943
sko81	0	0.40	0.4312	talent_10_100_8	3	0.51	0.3037
tai50a	0	0.58	0.2277	talent_10_100_8	4	-0.40	0.4375
lipa80a	0	0.25	0.6266	talent_10_100_8	5	0.73	0.0973

Table B.3: Pearson Product-Moment Correlations of P_{GB} comparing Ant System and GBAS varying α (NA (0/1)=Points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. *r*-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$, QAP is $\alpha = \frac{0.05}{8} = 0.0063$.)

Problem	β	r	P-Value	Problem	β	r	P-Value
gr17	0	0.78	0.0694	esc16c	0	0.74	0.0956
gr17	1	0.87	0.0255^{*B}	bur26f	0	0.91	0.0118^{*B}
gr17	2	0.76	0.0798	tai15b	0	0.89	0.0164^{*B}
gr17	3	0.53	0.2777	tai100b	0	0.80	0.0578
gr17	4	0.60	0.2122	talent_10_10_5	0	0.82	0.0437*
gr17	5	0.83	0.0412^{*B}	talent_10_10_5	1	0.86	0.0287^{*B}
bayg29	0	0.86	0.0277*	talent_10_10_5	2	0.39	0.4440
bayg29	1	1.00	< 0.0001*	talent_10_10_5	3	0.88	0.0221^{*B}
bayg29	2	0.99	0.0001*	talent_10_10_5	4	0.81	0.0515
bayg29	3	0.98	0.0005*	talent_10_10_5	5	0.81	0.0530
bayg29	4	0.93	0.0079^{*B}	talent_10_15_8	0	0.65	0.1647
bayg29	5	0.87	0.0257^{*B}	talent_10_15_8	1	0.86	0.0298^{*B}
brazil58	0	0.70	0.1226	talent_10_15_8	2	0.83	0.0432*
brazil58	1	0.97	0.0012*	talent_10_15_8	3	0.82	0.0458^{*B}
brazil58	2	0.98	0.0009*	talent_10_15_8	4	0.89	0.0177^{*B}
brazil58	3	0.99	0.0001*	talent_10_15_8	5	0.67	0.1428
brazil58	4	0.98	0.0006*	talent_10_20_5	0	0.85	0.0326^{*B}
brazil58	5	0.95	0.0043*	talent_10_20_5	1	0.71	0.1121
eil76	0	0.69	0.1300	talent_10_20_5	2	0.84	0.0342^{*B}
eil76	1	0.96	0.0020^{*B}	talent_10_20_5	3	0.87	0.0238^{*B}
eil76	2	0.97	0.0014*	talent_10_20_5	4	0.73	0.0994
eil76	3	0.97	0.0009*	talent_10_20_5	5	0.87	0.0243^{*B}
eil76	4	0.99	0.0003*	talent_10_30_4	0	0.83	0.0423*
eil76	5	1.00	< 0.0001*	talent_10_30_4	1	0.80	0.05369
kroA100	0	0.69	0.1305	talent_10_30_4	2	0.91	0.0112^{*B}
kroA100	1	0.96	0.0020^{*B}	talent_10_30_4	3	0.88	0.0213^{*B}
kroA100	2	0.98	0.0008*	talent_10_30_4	4	0.85	0.0315^{*B}
kroA100	3	0.95	0.0040^{*B}	talent_10_30_4	5	0.78	0.0694
kroA100	4	0.97	0.0014*	talent_10_100_8	0	0.42	0.4033
kroA100	5	0.96	0.0019^{*B}	talent_10_100_8	1	0.72	0.1093
had18	0	0.63	0.1805	talent_10_100_8	2	0.98	0.0005*
sko81	0	0.68	0.1348	talent_10_100_8	3	0.95	0.0040^{*B}
tai50a	0	0.85	0.0339^{*B}	talent_10_100_8	4	0.94	0.0049^{*B}
lipa80a	0	0.87	0.0231*	talent_10_100_8	5	0.95	0.0037^{*B}

Table B.4: Pearson Product-Moment Correlations of P_{GB} comparing MMAS and GBAS varying α. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. *r*-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$, QAP is $\alpha = \frac{0.05}{8} = 0.0063$.)

Problem	α	**r**	**P-Value**	**Problem**	α	**r**	**P-Value**
gr17	0	1.00	< 0.0001*	talent_10_10_5	0	0.92	0.0098^{*B}
gr17	1	1.00	< 0.0001*	talent_10_10_5	1	0.57	0.2344
gr17	2	0.99	< 0.0001*	talent_10_10_5	2	0.90	0.0157^{*B}
gr17	3	0.99	0.0002*	talent_10_10_5	3	0.90	0.0148^{*B}
gr17	4	0.99	< 0.0001*	talent_10_10_5	4	0.88	0.0195^{*B}
gr17	5	0.99	0.0001*	talent_10_10_5	5	0.87	0.0234^{*B}
bayg29	0	1.00	< 0.0001*	talent_10_15_8	0	0.99	< 0.0001*
bayg29	1	1.00	< 0.0001*	talent_10_15_8	1	0.91	0.0126^{*B}
bayg29	2	1.00	< 0.0001*	talent_10_15_8	2	0.51	0.3010
bayg29	3	1.00	< 0.0001*	talent_10_15_8	3	0.89	0.0166^{*B}
bayg29	4	1.00	< 0.0001*	talent_10_15_8	4	0.91	0.0119^{*B}
bayg29	5	0.99	< 0.0001*	talent_10_15_8	5	0.91	0.0108^{*B}
brazil58	0	1.00	< 0.0001*	talent_10_20_5	0	0.97	0.0018^{*B}
brazil58	1	1.00	< 0.0001*	talent_10_20_5	1	0.88	0.0221^{*B}
brazil58	2	1.00	< 0.0001*	talent_10_20_5	2	0.88	0.0222^{*B}
brazil58	3	1.00	< 0.0001*	talent_10_20_5	3	0.85	0.0312^{*B}
brazil58	4	0.99	0.0002*	talent_10_20_5	4	0.90	0.0146^{*B}
brazil58	5	0.99	0.0003*	talent_10_20_5	5	0.96	0.0022^{*B}
eil76	0	1.00	< 0.0001*	talent_10_30_4	0	0.99	0.0002*
eil76	1	1.00	< 0.0001*	talent_10_30_4	1	0.93	0.0076^{*B}
eil76	2	0.99	0.0002*	talent_10_30_4	2	0.95	0.0040^{*B}
eil76	3	0.98	0.0005*	talent_10_30_4	3	0.99	0.0003*
eil76	4	0.97	0.0014*	talent_10_30_4	4	0.95	0.0037^{*B}
eil76	5	0.97	0.0015*	talent_10_30_4	5	0.83	0.0424^{*B}
kroA100	0	1.00	< 0.0001*	talent_10_100_8	0	1.00	< 0.0001*
kroA100	1	1.00	< 0.0001*	talent_10_100_8	1	0.89	0.0169^{*B}
kroA100	2	0.99	< 0.0001*	talent_10_100_8	2	0.93	0.0062^{*B}
kroA100	3	0.99	0.0002*	talent_10_100_8	3	0.94	0.0059^{*B}
kroA100	4	0.99	0.0002*	talent_10_100_8	4	0.92	0.0083^{*B}
kroA100	5	0.99	0.0003*	talent_10_100_8	5	0.90	0.0148^{*B}

Table B.5: Pearson Product-Moment Correlations of P_{GB} comparing Ant System and GBAS varying β. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. *r*-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$.)

Problem	α	r	P-Value	Problem	α	r	P-Value
gr17	0	1.00	< 0.0001*	talent_10_10_5	0	0.91	0.0110^{*B}
gr17	1	0.96	0.0027^{*B}	talent_10_10_5	1	0.42	0.4000
gr17	2	0.96	0.0029^{*B}	talent_10_10_5	2	0.55	0.2600
gr17	3	0.99	0.0003*	talent_10_10_5	3	0.25	0.6400
gr17	4	0.99	0.0003*	talent_10_10_5	4	0.50	0.3200
gr17	5	0.99	0.0003*	talent_10_10_5	5	0.85	0.0310^{*B}
bayg29	0	1.00	< 0.0001*	talent_10_15_8	0	0.97	0.0010*
bayg29	1	0.99	0.0001*	talent_10_15_8	1	0.89	0.0180^{*B}
bayg29	2	0.99	0.0001*	talent_10_15_8	2	0.67	0.1400
bayg29	3	1.00	< 0.0001*	talent_10_15_8	3	0.85	0.0300^{*B}
bayg29	4	0.99	< 0.0001*	talent_10_15_8	4	0.46	0.3600
bayg29	5	0.99	0.0003*	talent_10_15_8	5	0.42	0.4000
brazil58	0	1.00	< 0.0001*	talent_10_20_5	0	0.94	0.0054^{*B}
brazil58	1	1.00	< 0.0001*	talent_10_20_5	1	0.64	0.1700
brazil58	2	0.98	0.0004*	talent_10_20_5	2	0.90	0.0160^{*B}
brazil58	3	0.99	0.0002*	talent_10_20_5	3	0.79	0.0600
brazil58	4	0.97	0.0010*	talent_10_20_5	4	0.89	0.0190^{*B}
brazil58	5	0.98	0.0009*	talent_10_20_5	5	0.96	0.0025^{*B}
eil76	0	1.00	< 0.0001*	talent_10_30_4	0	0.99	0.0002*
eil76	1	1.00	< 0.0001*	talent_10_30_4	1	0.86	0.0280^{*B}
eil76	2	0.98	0.0004*	talent_10_30_4	2	0.92	0.0100^{*B}
eil76	3	0.98	0.0009*	talent_10_30_4	3	0.97	0.0016*
eil76	4	0.95	0.0036^{*B}	talent_10_30_4	4	0.99	< 0.0001*
eil76	5	0.94	0.0049^{*B}	talent_10_30_4	5	0.95	0.0030^{*B}
kroA100	0	1.00	< 0.0001*	talent_10_100_8	0	1.00	< 0.0001*
kroA100	1	1.00	< 0.0001*	talent_10_100_8	1	0.98	0.0004*
kroA100	2	0.99	0.0002*	talent_10_100_8	2	0.99	0.0002*
kroA100	3	0.98	0.0004*	talent_10_100_8	3	0.99	0.0002*
kroA100	4	0.98	0.0007*	talent_10_100_8	4	0.98	0.0006*
kroA100	5	0.97	0.0014*	talent_10_100_8	5	0.98	0.0005*

Table B.6: Pearson Product-Moment Correlations of P_{GB} comparing MMAS and GBAS varying β. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. r-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$.)

Problem	β	r	P-Value	Problem	β	r	P-Value
gr17	0	0.02	0.9679	sko81	0	-0.15	0.7709
gr17	1	0.49	0.3237	tai100b	0	-0.96	0.0026*
gr17	2	0.84	0.0376^{*B}	tai15b	0	0.42	0.4054
gr17	3	0.85	0.0303^{*B}	tai50a	0	-0.28	0.5962
gr17	4	0.89	0.0179^{*B}	talent_10_10_5	0	0.95	0.0036^{*B}
gr17	5	0.57	0.2427	talent_10_10_5	1	0.98	0.0008*
bayg29	0	0.85	0.0313^{*B}	talent_10_10_5	2	0.96	0.0020^{*B}
bayg29	1	0.60	0.2080	talent_10_10_5	3	0.84	0.0354*
bayg29	2	0.83	0.0401^{*B}	talent_10_10_5	4	0.97	0.0014*
bayg29	3	0.78	0.0645	talent_10_10_5	5	0.98	0.0008*
bayg29	4	0.94	0.0061^{*B}	talent_10_15_8	0	0.96	0.0024^{*B}
bayg29	5	0.39	0.4460	talent_10_15_8	1	0.93	0.0072^{*B}
brazil58	0	0.40	0.4348	talent_10_15_8	2	0.64	0.1706
brazil58	1	0.62	0.1904	talent_10_15_8	3	0.81	0.0533
brazil58	2	0.54	0.2712	talent_10_15_8	4	0.75	0.0888
brazil58	3	-0.05	0.9255	talent_10_15_8	5	0.83	0.0412^{*B}
brazil58	4	-0.20	0.7074	talent_10_20_5	0	0.83	0.0425^{*B}
brazil58	5	0.68	0.1345	talent_10_20_5	1	0.99	0.0002*
eil76	0	0.58	0.2229	talent_10_20_5	2	0.39	0.4418
eil76	1	0.11	0.8425	talent_10_20_5	3	0.89	0.0165^{*B}
eil76	2	0.74	0.0938	talent_10_20_5	4	0.98	0.0007*
eil76	3	0.53	0.2785	talent_10_20_5	5	0.97	0.0018^{*B}
eil76	4	0.61	0.2001	talent_10_30_4	0	0.97	0.0016*
eil76	5	0.77	0.0727	talent_10_30_4	1	0.71	0.1119
kroA100	0	0.20	0.6992	talent_10_30_4	2	0.58	0.2292
kroA100	1	0.37	0.4745	talent_10_30_4	3	0.75	0.0869
kroA100	2	0.36	0.4856	talent_10_30_4	4	0.29	0.5764
kroA100	3	0.56	0.2445	talent_10_30_4	5	0.72	0.1086
kroA100	4	0.17	0.7525	talent_10_100_8	0	0.67	0.1449
kroA100	5	0.57	0.2331	talent_10_100_8	1	-0.73	0.0985
bur26f	0	0.14	0.7897	talent_10_100_8	2	0.59	0.2147
esc16c	0	0.53	0.2835	talent_10_100_8	3	0.31	0.5549
had18	0	0.71	0.1146	talent_10_100_8	4	-0.29	0.5817
lipa80a	0	0.07	0.8961	talent_10_100_8	5	-0.47	0.3485

Table B.7: Pearson Product-Moment Correlations of I_{GB} comparing Ant System and GBAS varying α. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. *r*-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$, QAP is $\alpha = \frac{0.05}{8} = 0.0063$.)

Problem	α	r	P-Value	Problem	α	r	P-Value
gr17	0	0.75	0.0891	talent_10_10_5	0	0.47	0.3464
gr17	1	0.67	0.1480	talent_10_10_5	1	0.43	0.3936
gr17	2	0.89	0.0160^{*B}	talent_10_10_5	2	0.87	0.0232^{*B}
gr17	3	0.97	0.0016*	talent_10_10_5	3	0.67	0.1428
gr17	4	0.95	0.0032^{*B}	talent_10_10_5	4	0.40	0.4259
gr17	5	0.87	0.0229^{*B}	talent_10_10_5	5	0.42	0.4103
bayg29	0	0.61	0.2029	talent_10_15_8	0	0.07	0.9003
bayg29	1	0.95	0.0036^{*B}	talent_10_15_8	1	-0.08	0.8806
bayg29	2	0.99	0.0002*	talent_10_15_8	2	0.45	0.3699
bayg29	3	0.96	0.0028^{*B}	talent_10_15_8	3	0.65	0.1614
bayg29	4	0.99	< 0.0001*	talent_10_15_8	4	-0.26	0.6189
bayg29	5	0.94	0.0056^{*B}	talent_10_15_8	5	-0.53	0.2746
brazil58	0	0.57	0.2359	talent_10_20_5	0	-0.08	0.8753
brazil58	1	1.00	< 0.0001*	talent_10_20_5	1	0.62	0.1880
brazil58	2	0.97	0.0010*	talent_10_20_5	2	0.44	0.3800
brazil58	3	0.97	0.0017*	talent_10_20_5	3	0.53	0.2742
brazil58	4	0.97	0.0015*	talent_10_20_5	4	0.19	0.7177
brazil58	5	0.56	0.2525	talent_10_20_5	5	0.56	0.2427
eil76	0	0.40	0.4276	talent_10_30_4	0	-0.56	0.2526
eil76	1	0.98	0.0004*	talent_10_30_4	1	0.81	0.0532
eil76	2	0.97	0.0011*	talent_10_30_4	2	0.32	0.5401
eil76	3	0.98	0.0008*	talent_10_30_4	3	-0.40	0.4288
eil76	4	0.99	0.0001*	talent_10_30_4	4	0.25	0.6349
eil76	5	0.87	0.0232^{*B}	talent_10_30_4	5	-0.75	0.0885
kroA100	0	-0.34	0.5138	talent_10_100_8	0	0.96	0.0027^{*B}
kroA100	1	0.99	0.0002*	talent_10_100_8	1	0.73	0.0975
kroA100	2	0.97	0.0010*	talent_10_100_8	2	0.86	0.0271^{*B}
kroA100	3	0.98	0.0006*	talent_10_100_8	3	0.19	0.7241
kroA100	4	1.00	< 0.0001*	talent_10_100_8	4	0.14	0.7891
kroA100	5	0.79	0.0606	talent_10_100_8	5	0.49	0.3276

Table B.8: Pearson Product-Moment Correlations of I_{GB} comparing Ant System and GBAS varying β. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. *r*-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$)

Problem	β	r	P-Value	Problem	β	r	P-Value
gr17	0	0.32	0.5301	sko81	0	0.68	0.1344
gr17	1	0.66	0.1531	tai100b	0	0.73	0.0963
gr17	2	0.96	0.0023^{*B}	tai15b	0	0.56	0.2463
gr17	3	0.99	0.0002*	tai50a	0	0.59	0.2211
gr17	4	0.99	0.0002*	talent_10_10_5	0	0.53	0.2809
gr17	5	0.99	< 0.0001*	talent_10_10_5	1	0.98	0.0007*
bayg29	0	0.62	0.1890	talent_10_10_5	2	0.91	0.0130^{*B}
bayg29	1	0.58	0.2325	talent_10_10_5	3	0.86	0.0267^{*B}
bayg29	2	0.32	0.5331	talent_10_10_5	4	0.45	0.3694
bayg29	3	0.76	0.0800	talent_10_10_5	5	0.82	0.0440^{*B}
bayg29	4	0.84	0.0371^{*B}	talent_10_15_8	0	0.59	0.2133
bayg29	5	0.94	0.0050^{*B}	talent_10_15_8	1	0.54	0.2656
brazil58	0	0.61	0.2018	talent_10_15_8	2	0.63	0.1774
brazil58	1	0.77	0.0737	talent_10_15_8	3	0.65	0.1631
brazil58	2	0.74	0.0895	talent_10_15_8	4	0.78	0.0648
brazil58	3	0.62	0.1909	talent_10_15_8	5	0.55	0.2572
brazil58	4	0.67	0.1413	talent_10_20_5	0	0.52	0.2850
brazil58	5	0.81	0.0503	talent_10_20_5	1	0.93	0.0080^{*B}
eil76	0	0.76	0.0768	talent_10_20_5	2	0.52	0.2930
eil76	1	0.85	0.0318^{*B}	talent_10_20_5	3	0.90	0.0157^{*B}
eil76	2	0.70	0.1201	talent_10_20_5	4	0.42	0.4084
eil76	3	0.66	0.1550	talent_10_20_5	5	0.82	0.0479^{*B}
eil76	4	0.55	0.2593	talent_10_30_4	0	0.63	0.1762
eil76	5	0.69	0.1269	talent_10_30_4	1	0.66	0.1565
kroA100	0	0.82	0.0480^{*B}	talent_10_30_4	2	0.53	0.2841
kroA100	1	0.78	0.0665	talent_10_30_4	3	0.70	0.1217
kroA100	2	0.49	0.3249	talent_10_30_4	4	0.52	0.2971
kroA100	3	0.48	0.3404	talent_10_30_4	5	0.71	0.1119
kroA100	4	0.58	0.2257	talent_10_100_8	0	0.80	0.0535
kroA100	5	0.70	0.1188	talent_10_100_8	1	0.78	0.0700
bur26f	0	0.59	0.2195	talent_10_100_8	2	0.65	0.1629
esc16c	0	0.39	0.4499	talent_10_100_8	3	0.49	0.3221
had18	0	0.47	0.3414	talent_10_100_8	4	0.65	0.1645
lipa80a	0	0.74	0.0935	talent_10_100_8	5	0.97	0.0010*

Table B.9: Pearson Product-Moment Correlations of I_{GB} comparing MMAS and GBAS varying α. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. r-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$, QAP is $\alpha = \frac{0.05}{8} = 0.0063$.)

Problem	α	r	P-Value	Problem	α	r	P-Value
gr17	0	0.00	0.9953	talent_10_10_5	0	0.77	0.0705
gr17	1	0.86	0.0267^{*B}	talent_10_10_5	1	0.01	0.9804
gr17	2	0.99	0.0002*	talent_10_10_5	2	-0.86	0.0263*
gr17	3	0.97	0.0017*	talent_10_10_5	3	0.73	0.1009
gr17	4	0.78	0.0660	talent_10_10_5	4	0.73	0.0962
gr17	5	0.92	0.0091^{*B}	talent_10_10_5	5	0.17	0.7474
bayg29	0	-0.51	0.3046	talent_10_15_8	0	0.57	0.2374
bayg29	1	0.91	0.0118^{*B}	talent_10_15_8	1	0.48	0.3309
bayg29	2	0.96	0.0027^{*B}	talent_10_15_8	2	0.19	0.7169
bayg29	3	0.98	0.0009*	talent_10_15_8	3	0.72	0.1070
bayg29	4	0.97	0.0011*	talent_10_15_8	4	0.42	0.4130
bayg29	5	0.90	0.0147^{*B}	talent_10_15_8	5	-0.52	0.2943
brazil58	0	-0.08	0.8750	talent_10_20_5	0	-0.41	0.4214
brazil58	1	0.65	0.1661	talent_10_20_5	1	-0.37	0.4655
brazil58	2	0.94	0.0049^{*B}	talent_10_20_5	2	-0.58	0.2266
brazil58	3	0.99	< 0.0001*	talent_10_20_5	3	0.26	0.6238
brazil58	4	0.98	0.0008*	talent_10_20_5	4	-0.19	0.7226
brazil58	5	0.39	0.4420	talent_10_20_5	5	0.33	0.5226
eil76	0	-0.59	0.2130	talent_10_30_4	0	0.47	0.3488
eil76	1	0.86	0.0294^{*B}	talent_10_30_4	1	0.11	0.8358
eil76	2	0.99	0.0002*	talent_10_30_4	2	0.33	0.5271
eil76	3	0.99	< 0.0001*	talent_10_30_4	3	-0.34	0.5122
eil76	4	0.96	0.0029^{*B}	talent_10_30_4	4	-0.30	0.5696
eil76	5	0.72	0.1068	talent_10_30_4	5	-0.82	0.0449^{*B}
kroA100	0	-0.46	0.3575	talent_10_100_8	0	-0.02	0.9707
kroA100	1	0.81	0.0527	talent_10_100_8	1	0.78	0.0666
kroA100	2	1.00	< 0.0001*	talent_10_100_8	2	0.00	0.9943
kroA100	3	1.00	< 0.0001*	talent_10_100_8	3	-0.76	0.0803
kroA100	4	0.92	0.0936	talent_10_100_8	4	-0.53	0.2837
kroA100	5	0.51	0.2996	talent_10_100_8	5	-0.57	0.2326

Table B.10: Pearson Product-Moment Correlations of I_{GB} comparing MMAS and GBAS varying β. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. r-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$.)

Problem	N	ρ_{alg}	AS		MMAS	
			r	p-value	r	p-value
gr17	17	0.70	0.98	< 0.0001*	0.98	< 0.0001*
gr17	17	0.90	0.98	< 0.0001*	0.97	< 0.0001*
gr17	17	0.95	0.96	< 0.0001*	0.98	< 0.0001*
gr17	17	0.99	0.93	< 0.0001*	0.93	< 0.0001*
bayg29	29	0.70	0.97	< 0.0001*	0.98	< 0.0001*
bayg29	29	0.90	0.98	< 0.0001*	0.99	< 0.0001*
bayg29	29	0.95	0.98	< 0.0001*	0.98	< 0.0001*
bayg29	29	0.99	0.92	< 0.0001*	0.95	< 0.0001*
brazil58	58	0.70	0.99	< 0.0001*	0.98	< 0.0001*
brazil58	58	0.90	0.99	< 0.0001*	0.99	< 0.0001*
brazil58	58	0.95	0.99	< 0.0001*	0.99	< 0.0001*
brazil58	58	0.99	0.95	< 0.0001*	0.99	< 0.0001*
eil76	76	0.70	0.99	< 0.0001*	0.99	< 0.0001*
eil76	76	0.90	0.99	< 0.0001*	0.99	< 0.0001*
eil76	76	0.95	0.99	< 0.0001*	0.99	< 0.0001*
eil76	76	0.99	0.97	< 0.0001*	0.98	< 0.0001*
kroA100	100	0.70	0.99	< 0.0001*	0.99	< 0.0001*
kroA100	100	0.90	0.99	< 0.0001*	0.99	< 0.0001*
kroA100	100	0.95	0.99	< 0.0001*	0.98	< 0.0001*
kroA100	100	0.99	0.97	< 0.0001*	0.99	< 0.0001*
had18	18	0.70	0.93	< 0.0001*	0.92	< 0.0001*
had18	18	0.90	0.87	< 0.0001*	0.94	< 0.0001*
had18	18	0.95	0.87	< 0.0001*	0.85	< 0.0001*
had18	18	0.99	0.88	< 0.0001*	0.91	< 0.0001*
sko81	81	0.70	0.93	< 0.0001*	0.95	< 0.0001*
sko81	81	0.90	0.92	< 0.0001*	0.94	< 0.0001*
sko81	81	0.95	0.95	< 0.0001*	0.94	< 0.0001*
sko81	81	0.99	0.96	< 0.0001*	0.94	< 0.0001*
tai50a	50	0.70	0.92	< 0.0001*	0.90	< 0.0001*
tai50a	50	0.90	0.92	< 0.0001*	0.90	< 0.0001*
tai50a	50	0.95	0.88	< 0.0001*	0.88	< 0.0001*
tai50a	50	0.99	0.86	< 0.0001*	0.87	< 0.0001*
lipa80a	80	0.70	0.97	< 0.0001*	0.95	< 0.0001*
lipa80a	80	0.90	0.93	< 0.0001*	0.95	< 0.0001*
lipa80a	80	0.95	0.94	< 0.0001*	0.93	< 0.0001*
lipa80a	80	0.99	0.96	< 0.0001*	0.90	< 0.0001*

Table B.11: Correlations for P_{GB} between AS/MMAS and GBAS, and their associated p-values. (*=Significant correlations given to 4 decimal places, *r* values given to 2 decimal places. NA (0/1)=Those points where all values are equal and therefore have no standard deviation. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\frac{0.05}{72} = 0.0007$.)

Problem	N	ρ_{alg}	AS		MMAS	
			r	p-value	r	p-value
esc16c	16	0.70	0.88	< 0.0001*	0.93	< 0.0001*
esc16c	16	0.90	0.93	< 0.0001*	0.84	< 0.0001*
esc16c	16	0.95	0.83	< 0.0001*	0.93	< 0.0001*
esc16c	16	0.99	0.90	< 0.0001*	0.90	< 0.0001*
bur26f	26	0.70	0.93	< 0.0001*	0.96	< 0.0001*
bur26f	26	0.90	0.91	< 0.0001*	0.92	< 0.0001*
bur26f	26	0.95	0.93	< 0.0001*	0.95	< 0.0001*
bur26f	26	0.99	0.94	< 0.0001*	0.96	< 0.0001*
tai15b	15	0.70	0.60	0.0136*	0.62	0.0102*
tai15b	15	0.90	0.94	< 0.0001*	0.95	< 0.0001*
tai15b	15	0.95	0.90	< 0.0001*	0.93	< 0.0001*
tai15b	15	0.99	0.94	< 0.0001*	0.95	< 0.0001*
tai100b	100	0.70	0.90	< 0.0001*	0.94	< 0.0001*
tai100b	100	0.90	0.93	< 0.0001*	0.88	< 0.0001*
tai100b	100	0.95	0.93	< 0.0001*	0.88	< 0.0001*
tai100b	100	0.99	0.94	< 0.0001*	0.89	< 0.0001*
talent_10_10_5	10	0.70	0.96	< 0.0001*	0.94	< 0.0001*
talent_10_10_5	10	0.90	0.90	0.0001*	0.86	0.0007*
talent_10_10_5	10	0.95	0.93	< 0.0001*	0.97	< 0.0001*
talent_10_10_5	10	0.99	0.87	0.0004*	0.84	0.0011*
talent_10_15_8	15	0.70	0.87	< 0.0001*	0.82	< 0.0001*
talent_10_15_8	15	0.90	0.93	< 0.0001*	0.90	< 0.0001*
talent_10_15_8	15	0.95	0.90	< 0.0001*	0.93	< 0.0001*
talent_10_15_8	15	0.99	0.94	< 0.0001*	0.88	< 0.0001*
talent_10_20_5	20	0.70	0.92	< 0.0001*	0.93	< 0.0001*
talent_10_20_5	20	0.90	0.93	< 0.0001*	0.95	< 0.0001*
talent_10_20_5	20	0.95	0.91	< 0.0001*	0.91	< 0.0001*
talent_10_20_5	20	0.99	0.91	< 0.0001*	0.93	< 0.0001*
talent_10_30_4	30	0.70	0.95	< 0.0001*	0.94	< 0.0001*
talent_10_30_4	30	0.90	0.94	< 0.0001*	0.95	< 0.0001*
talent_10_30_4	30	0.95	0.88	< 0.0001*	0.93	< 0.0001*
talent_10_30_4	30	0.99	0.91	< 0.0001*	0.94	< 0.0001*
talent_10_100_8	100	0.70	0.93	< 0.0001*	0.93	< 0.0001*
talent_10_100_8	100	0.90	0.93	< 0.0001*	0.96	< 0.0001*
talent_10_100_8	100	0.95	0.90	< 0.0001*	0.98	< 0.0001*
talent_10_100_8	100	0.99	0.95	< 0.0001*	0.91	< 0.0001*

Table B.12: *Continued.* Correlations for P_{GB} between AS/MMAS and GBAS, and their associated p-values. (*=Significant correlations given to 4 decimal places, *r* values given to 2 decimal places. NA (0/1)=Those points where all values are equal and therefore have no standard deviation. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\frac{0.05}{72} = 0.0007$.)

Problem	N	ρ_{alg}	**AS**		**MMAS**	
			r	p-value	r	p-value
gr17	17	0.70	0.75	$0.0003^{\star B}$	-0.14	0.5798
gr17	17	0.90	0.37	0.1291	-0.49	$0.0382^{\star B}$
gr17	17	0.95	0.16	0.5230	-0.18	0.4797
gr17	17	0.99	-0.06	0.8092	0.10	0.6789
bayg29	29	0.70	0.80	< 0.0001*	0.26	0.1634
bayg29	29	0.90	0.40	$0.0303^{\star B}$	0.04	0.8417
bayg29	29	0.95	0.29	0.1266	0.24	0.1932
bayg29	29	0.99	-0.34	0.0620	-0.10	0.6092
brazil58	58	0.70	0.89	< 0.0001*	0.51	0.0321*
brazil58	58	0.90	0.82	< 0.0001*	0.49	0.0370*
brazil58	58	0.95	0.49	$0.0404^{\star B}$	-0.16	0.5161
brazil58	58	0.99	-0.08	0.7396	0.10	0.7037
eil76	76	0.70	0.84	< 0.0001*	0.59	$0.0032^{\star B}$
eil76	76	0.90	0.84	< 0.0001*	0.83	< 0.0001*
eil76	76	0.95	0.40	0.0571	0.09	0.6967
eil76	76	0.99	0.38	0.0751	0.31	0.1481
kroA100	100	0.70	0.93	< 0.0001*	0.56	$0.0015^{\star B}$
kroA100	100	0.90	0.75	< 0.0001*	0.45	$0.0141^{\star B}$
kroA100	100	0.95	0.45	$0.0152^{\star B}$	0.35	0.0668
kroA100	100	0.99	-0.14	0.4568	0.28	0.1357
had18	18	0.70	0.10	0.6914	0.21	0.3880
had18	18	0.90	-0.04	0.8678	-0.00	0.9900
had18	18	0.95	0.65	$0.0028^{\star B}$	0.19	0.4317
had18	18	0.99	0.50	$0.0310^{\star B}$	0.18	0.4759
sko81	81	0.70	0.59	0.0022*	0.24	0.2489
sko81	81	0.90	0.74	< 0.0001*	0.45	$0.0269^{\star B}$
sko81	81	0.95	0.55	$0.0057^{\star B}$	0.45	$0.0287^{\star B}$
sko81	81	0.99	0.14	0.5090	0.18	0.4082
tai50a	50	0.70	0.26	0.0652	0.02	0.8704
tai50a	50	0.90	0.49	0.0003*	-0.01	0.9462
tai50a	50	0.95	0.25	0.0738	0.42	$0.0021^{\star B}$
tai50a	50	0.99	0.13	0.3592	-0.09	0.5301
lipa80a	80	0.70	0.59	$0.0022^{\star B}$	0.42	$0.0394^{\star B}$
lipa80a	80	0.90	0.58	$0.0030^{\star B}$	0.29	0.1640
lipa80a	80	0.95	0.32	0.1327	0.44	$0.0314^{\star B}$
lipa80a	80	0.99	-0.08	0.6978	0.20	0.3369

Table B.13: Correlations for I_{GB} between AS/MMAS and GBAS, and their associated p-values. (*=Significant correlations given to 4 decimal places, *r* values given to 2 decimal places.NA (0/1)=Those points where all values are equal and therefore have no standard deviation. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\frac{0.05}{72} = 0.0007$.)

Problem	N	ρ_{alg}	AS		MMAS	
			r	p-value	r	p-value
esc16c	16	0.70	0.30	0.2440	0.40	0.1087
esc16c	16	0.90	-0.16	0.5290	-0.47	0.0586
esc16c	16	0.95	0.39	0.1186	-0.07	0.7933
esc16c	16	0.99	-0.24	0.3441	0.15	0.5627
bur26f	26	0.70	0.49	$0.0095^{\star B}$	0.41	$0.0330^{\star B}$
bur26f	26	0.90	0.54	$0.0034^{\star B}$	0.13	0.5209
bur26f	26	0.95	0.37	0.0573	0.36	0.0657
bur26f	26	0.99	0.16	0.4176	0.29	0.1413
tai15b	15	0.70	0.33	0.2110	0.07	0.8018
tai15b	15	0.90	0.27	0.3183	-0.02	0.9272
tai15b	15	0.95	0.47	0.0659	-0.06	0.8341
tai15b	15	0.99	-0.06	0.8113	0.56	$0.0246^{\star B}$
tai100b	100	0.70	0.41	$0.0274^{\star B}$	0.15	0.4303
tai100b	100	0.90	0.29	0.1307	-0.01	0.9507
tai100b	100	0.95	-0.15	0.4458	0.37	$0.0463^{\star B}$
tai100b	100	0.99	-0.22	0.2582	0.32	0.0916
talent_10_10_5	10	0.70	0.13	0.6983	0.37	0.2673
talent_10_10_5	10	0.90	-0.17	0.6149	-0.50	0.1197
talent_10_10_5	10	0.95	-0.61	$0.0443^{\star B}$	-0.08	0.8180
talent_10_10_5	10	0.99	-0.32	0.3392	0.74	$0.0086^{\star B}$
talent_10_15_8	15	0.70	0.56	$0.0231^{\star B}$	0.13	0.6325
talent_10_15_8	15	0.90	0.56	$0.0236^{\star B}$	0.16	0.5472
talent_10_15_8	15	0.95	0.39	0.1340	0.32	0.2218
talent_10_15_8	15	0.99	-0.12	0.6511	0.04	0.8862
talent_10_20_5	20	0.70	0.51	$0.0170^{\star B}$	-0.21	0.3571
talent_10_20_5	20	0.90	0.20	0.3957	-0.22	0.3590
talent_10_20_5	20	0.95	0.41	0.0629	0.29	0.2031
talent_10_20_5	20	0.99	0.26	0.2471	-0.00	0.9915
talent_10_30_4	30	0.70	0.55	$0.0012^{\star B}$	0.29	0.1123
talent_10_30_4	30	0.90	0.53	$0.0022^{\star B}$	-0.04	0.8237
talent_10_30_4	30	0.95	0.26	0.1573	0.19	0.3064
talent_10_30_4	30	0.99	-0.17	0.3536	0.31	0.0892
talent_10_100_8	100	0.70	0.77	$< 0.0001^{\star}$	0.36	0.0572
talent_10_100_8	100	0.90	0.74	$< 0.0001^{\star}$	0.39	$0.0356^{\star B}$
talent_10_100_8	100	0.95	0.67	$< 0.0001^{\star}$	0.56	$0.0016^{\star B}$
talent_10_100_8	100	0.99	-0.22	0.2613	0.27	0.1638

Table B.14: *Continued.* Correlations for I_{GB} between AS/MMAS and GBAS, and their associated p-values. (*=Significant correlations given to 4 decimal places, *r* values given to 2 decimal places. NA (0/1)=Those points where all values are equal and therefore have no standard deviation. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected α is $\frac{0.05}{72} = 0.0007$.)

Appendix C

Tables for Chapter 7

	Problem	AS,GBAS Correlation		MMAS,GBAS Correlation	
ρ_{alg} **Range**		$(0,0.9]$	$[0.9,1)$	$(0,0.9]$	$[0.9,1)$
TSP					
	kroA150	0.9388*	0.1760	-0.2497	0.4204
	rat195	0.9859*	-0.2553	-0.4106	0.0420
	gr202	0.8893*	0.1215	-0.3022	0.2510
	att532	0.9328*	0.3880	0.4783	0.5124
	rat575	0.9642*	0.7170	0.5444	0.7113
QAP					
UIGD	nug20	0.8951*	0.6649	-0.0771	-0.5809
UIGD	sko100a	0.9113*	0.9661'	0.6528	0.9829'
URGI	lipa40a	0.8030^{*B}	-0.1009	0.5389	0.1521
URGI	tho150	0.8932^{*}	$0.7697^{'B}$	0.7421^{*B}	0.9464'
RLI	esc16g	NA (1)*	NA (1)'	NA (1)*	NA (1)'
RLI	bur26d	0.6172	0.2170	0.7702^{*B}	-0.5039
RLLI	tai35b	0.9826*	0.4250	0.4392	0.3984
RLLI	tai60b	0.8305*	0.5777	0.8512*	0.9101'
TS					
	talent_10_70_8	0.6361	$0.7894^{'B}$	0.4290	$0.8520^{'B}$
	talent_10_100_5	0.8603*	0.9279'	0.6520	0.9497'
	talent_10_150_4	0.7828^{*B}	0.9200'	0.6425	0.9326'
	talent_10_175_4	0.5073	0.9303'	0.8546*	0.9584'
	talent_10_200_5	0.3875	0.9403'	0.7190^{*B}	0.9704'

Table C.1: Summary of results for the correlations for P_{GB} between GBAS and AS/MMAS varying parameter ρ_{alg}. Correlations calculated using the Pearson Product-Moment Correlation. (* = Significant correlation for $(0,0.9]$, degrees of freedom$= 9-2=7$, 0.05 significance$=0.669$, ' = Significant Correlation for [0.9,1), degrees of freedom$= 6-2=4$, 0.05 significance$=0.729$, significance values taken from [Coolican, 1994]. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are: TSP,TS is $\alpha = \frac{0.05}{5} = 0.01$, and QAP is $\alpha = \frac{0.05}{8} = 0.0063$.)

	Problem	AS,GBAS Correlation		MMAS,GBAS Correlation	
ρ_{alg} **Range**		$(0, 0.9]$	$[0.9, 1)$	$(0, 0.9]$	$[0.9, 1)$
TSP					
	kroA150	0.9928*	0.0724	-0.2937	0.5763
	rat195	0.9981*	0.2843	-0.3761	$0.7674^{'B}$
	gr202	0.9752*	0.1305	-0.3879	$0.8186^{'B}$
	att532	0.9974*	-0.1185	0.0046	0.4816
	rat575	0.9952*	-0.2347	0.0339	$0.7542^{'B}$
QAP					
UIGD	nug20	0.9827*	0.9035'	-0.5082	0.5085
UIGD	sko100a	0.9985*	0.0698	0.0934	-0.2335
URGI	lipa40a	0.9877*	0.5926	0.0029	-0.0975
URGI	tho150	0.9989*	0.4817	0.1396	0.5999
RLI	esc16g	0.1223	0.3000	0.2925	-0.2282
RLI	bur26d	0.9919*	0.0426	0.0865	0.0180
RLLI	tai35b	0.9933*	-0.4222	0.2022	$0.7692^{'B}$
RLLI	tai60b	0.9954*	-0.5683	0.1238	-0.2550
TS					
	talent_10_70_8	0.9976*	0.2161	-0.2206	0.4145
	talent_10_100_5	0.9984*	0.4916	0.4160	0.1642
	talent_10_150_4	0.9966*	0.1336	0.3301	0.0525
	talent_10_175_4	0.9990*	0.2379	0.3843	0.5765
	talent_10_200_5	0.9996*	0.1147	0.5871	0.5154

Table C.2: Summary of results for the correlation for I_{GB} between AS/MMAS and GBAS varying parameter ρ_{alg}. Correlations calculated using the Pearson Product-Moment Correlation. (* = Significant correlation for $(0, 0.9]$, degrees of freedom$= 9 - 2 = 7$, 0.05 significance$=0.669$, '= Significant Correlation for [0.9,1), degrees of freedom$= 6 - 2 = 4$, 0.05 significance$=0.729$, significance values taken from [Coolican, 1994]. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are: TSP,TS is $\alpha = \frac{0.05}{5} = 0.01$, and QAP is $\alpha = \frac{0.05}{8} = 0.0063$.)

Problem	β	r	P-Value	Problem	β	r	P-Value
kroA150	0	-0.07	0.8912	esc16g	0	NA (1)	NA (0)*
kroA150	1	0.23	0.6644	bur26d	0	0.55	0.2552
kroA150	2	-0.09	0.8619	tai35b	0	-0.21	0.6933
kroA150	3	0.38	0.4560	tai60b	0	-0.20	0.7111
kroA150	4	0.11	0.8381	talent_10_70_8	0	0.91	$0.0109^{\star B}$
kroA150	5	-0.25	0.6369	talent_10_70_8	1	0.94	0.0052*
rat195	0	0.51	0.3012	talent_10_70_8	2	0.96	0.0026*
rat195	1	0.33	0.5294	talent_10_70_8	3	0.95	0.0037*
rat195	2	0.07	0.8934	talent_10_70_8	4	0.82	$0.0448^{\star B}$
rat195	3	-0.05	0.9302	talent_10_70_8	5	0.90	$0.0140^{\star B}$
rat195	4	0.19	0.7170	talent_10_100_5	0	0.99	0.0001*
rat195	5	-0.02	0.9679	talent_10_100_5	1	1.00	< 0.0001*
gr202	0	0.23	0.6626	talent_10_100_5	2	0.96	0.0020*
gr202	1	0.01	0.9818	talent_10_100_5	3	0.95	0.0031*
gr202	2	0.21	0.6893	talent_10_100_5	4	0.88	$0.0213^{\star B}$
gr202	3	0.29	0.5789	talent_10_100_5	5	0.96	0.0028*
gr202	4	0.39	0.4464	talent_10_150_4	0	0.96	0.0024*
gr202	5	0.08	0.8860	talent_10_150_4	1	0.91	$0.0124^{\star B}$
att532	0	0.18	0.7287	talent_10_150_4	2	0.93	$0.0081^{\star B}$
att532	1	0.21	0.6845	talent_10_150_4	3	0.95	0.0039*
att532	2	0.26	0.6229	talent_10_150_4	4	0.91	$0.0129^{\star B}$
att532	3	0.61	0.1939	talent_10_150_4	5	0.96	0.0027*
att532	4	0.30	0.5663	talent_10_175_4	0	0.88	$0.0193^{\star B}$
att532	5	0.30	0.5623	talent_10_175_4	1	0.88	$0.0193^{\star B}$
rat575	0	0.40	0.4271	talent_10_175_4	2	0.93	$0.0069^{\star B}$
rat575	1	0.61	0.2011	talent_10_175_4	3	0.90	$0.0141^{\star B}$
rat575	2	0.49	0.3217	talent_10_175_4	4	0.93	$0.0080^{\star B}$
rat575	3	0.46	0.3541	talent_10_175_4	5	0.90	$0.0146^{\star B}$
rat575	4	0.64	0.1737	talent_10_200_5	0	0.90	$0.0135^{\star B}$
rat575	5	0.58	0.2301	talent_10_200_5	1	0.91	$0.0109^{\star B}$
nug20	5	0.78	0.0671	talent_10_200_5	2	0.93	$0.0069^{\star B}$
sko100a	0	0.68	0.1402	talent_10_200_5	3	0.91	$0.0110^{\star B}$
lipa40a	0	-0.20	0.7062	talent_10_200_5	4	0.92	$0.0099^{\star B}$
tho150	0	0.37	0.4765	talent_10_200_5	5	0.95	0.0035*

Table C.3: Pearson Product-Moment Correlations of P_{GB} comparing AS and GBAS varying α. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. *r*-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$, QAP is $\alpha = \frac{0.05}{8} = 0.0063$.)

Problem	α	r	P-Value	Problem	α	r	P-Value
kroA150	0	0.32	0.5323	talent_10_70_8	0	0.43	0.3987
kroA150	1	0.13	0.8030	talent_10_70_8	1	0.74	0.0938
kroA150	2	0.46	0.3543	talent_10_70_8	2	0.33	0.5172
kroA150	3	-0.57	0.2358	talent_10_70_8	3	0.40	0.4342
kroA150	4	-0.07	0.8952	talent_10_70_8	4	0.24	0.6401
kroA150	5	0.40	0.4366	talent_10_70_8	5	-0.20	0.7097
rat195	0	-0.13	0.8126	talent_10_100_5	0	0.75	0.0886
rat195	1	-0.27	0.6041	talent_10_100_5	1	0.93	0.0079^{*B}
rat195	2	0.61	0.2022	talent_10_100_5	2	0.40	0.4355
rat195	3	0.51	0.2967	talent_10_100_5	3	-0.48	0.3396
rat195	4	0.58	0.2241	talent_10_100_5	4	0.77	0.0726
rat195	5	0.11	0.8297	talent_10_100_5	5	0.66	0.1532
gr202	0	0.60	0.2084	talent_10_150_4	0	0.90	0.0142^{*B}
gr202	1	-0.67	0.1461	talent_10_150_4	1	0.96	0.0025*
gr202	2	-0.63	0.1818	talent_10_150_4	2	0.79	0.0612
gr202	3	0.62	0.1901	talent_10_150_4	3	-0.05	0.9311
gr202	4	0.37	0.4764	talent_10_150_4	4	-0.76	0.0789
gr202	5	-0.14	0.7891	talent_10_150_4	5	0.22	0.6792
att532	0	-0.08	0.8784	talent_10_175_4	0	0.52	0.2884
att532	1	-0.26	0.6177	talent_10_175_4	1	0.99	0.0001*
att532	2	-0.24	0.6439	talent_10_175_4	2	0.87	0.0249^{*B}
att532	3	0.71	0.1151	talent_10_175_4	3	0.54	0.2695
att532	4	-0.83	0.0431^{*B}	talent_10_175_4	4	-0.48	0.3309
att532	5	-0.66	0.1514	talent_10_175_4	5	-0.44	0.3810
rat575	0	0.70	0.1203	talent_10_200_5	0	0.72	0.1065
rat575	1	0.90	0.0145^{*B}	talent_10_200_5	1	0.98	0.0004*
rat575	2	0.88	0.0213^{*B}	talent_10_200_5	2	0.88	0.0215^{*B}
rat575	3	0.76	0.0818	talent_10_200_5	3	0.77	0.0742
rat575	4	0.78	0.0687	talent_10_200_5	4	-0.00	0.9986
rat575	5	0.61	0.2003	talent_10_200_5	5	-0.75	0.0840

Table C.4: Pearson Product-Moment Correlations of P_{GB} comparing AS and GBAS varying β. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. r-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$.)

Problem	β	r	P-Value	Problem	β	r	P-Value
kroA150	0	0.91	$0.0114^{\star B}$	esc16g	0	NA (1)	NA (0)*
kroA150	1	0.98	0.0006*	bur26d	0	0.52	0.2869
kroA150	2	0.79	0.0643	tai35b	0	0.60	0.2075
kroA150	3	0.64	0.1751	tai60b	0	0.23	0.6588
kroA150	4	0.50	0.3161	talent_10_70_8	0	0.90	$0.0134^{\star B}$
kroA150	5	0.82	$0.0454^{\star B}$	talent_10_70_8	1	0.97	0.0013*
rat195	0	0.91	$0.0115^{\star B}$	talent_10_70_8	2	0.78	0.0668
rat195	1	0.62	0.1899	talent_10_70_8	3	0.78	0.0667
rat195	2	0.37	0.4677	talent_10_70_8	4	0.70	0.1202
rat195	3	0.49	0.3219	talent_10_70_8	5	0.72	0.1073
rat195	4	0.73	0.0988	talent_10_100_5	0	1.00	< 0.0001*
rat195	5	0.53	0.2759	talent_10_100_5	1	0.99	< 0.0001*
gr202	0	0.66	0.1568	talent_10_100_5	2	0.85	$0.0337^{\star B}$
gr202	1	0.60	0.2067	talent_10_100_5	3	0.78	0.0667
gr202	2	0.64	0.1715	talent_10_100_5	4	0.84	$0.0373^{\star B}$
gr202	3	0.58	0.2294	talent_10_100_5	5	0.88	$0.0207^{\star B}$
gr202	4	0.62	0.1889	talent_10_150_4	0	0.99	0.0002*
gr202	5	0.64	0.1690	talent_10_150_4	1	0.96	0.0020*
att532	0	0.46	0.3538	talent_10_150_4	2	0.87	$0.0254^{\star B}$
att532	1	0.27	0.6002	talent_10_150_4	3	0.93	$0.0079^{\star B}$
att532	2	0.49	0.3235	talent_10_150_4	4	0.86	$0.0270^{\star B}$
att532	3	0.62	0.1872	talent_10_150_4	5	0.89	$0.0162^{\star B}$
att532	4	0.58	0.2237	talent_10_175_4	0	0.95	0.0041*
att532	5	0.46	0.3541	talent_10_175_4	1	0.96	0.0020*
rat575	0	0.56	0.2513	talent_10_175_4	2	0.91	$0.0130^{\star B}$
rat575	1	0.47	0.3430	talent_10_175_4	3	0.94	0.0056*
rat575	2	0.51	0.3026	talent_10_175_4	4	0.92	$0.0096^{\star B}$
rat575	3	0.63	0.1793	talent_10_175_4	5	0.90	$0.0134^{\star B}$
rat575	4	0.67	0.1486	talent_10_200_5	0	0.97	0.0014*
rat575	5	0.69	0.1295	talent_10_200_5	1	0.98	0.0008*
nug20	5	0.73	0.1000	talent_10_200_5	2	0.93	$0.0075^{\star B}$
sko100a	0	0.66	0.1543	talent_10_200_5	3	0.95	0.0038*
lipa40a	0	0.54	0.2647	talent_10_200_5	4	0.93	$0.0074^{\star B}$
tho150	0	0.84	$0.0342^{\star B}$	talent_10_200_5	5	0.94	0.0061*

Table C.5: Pearson Product-Moment Correlations of P_{GB} comparing MMAS and GBAS varying α. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. *r*-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$, QAP is $\alpha = \frac{0.05}{8} = 0.0063$.)

Problem	α	r	P-Value	Problem	α	r	P-Value
kroA150	0	0.20	0.7074	talent_10_70_8	0	0.73	0.0994
kroA150	1	-0.02	0.9700	talent_10_70_8	1	0.52	0.2886
kroA150	2	0.40	0.4277	talent_10_70_8	2	-0.12	0.8227
kroA150	3	-0.07	0.8967	talent_10_70_8	3	0.52	0.2883
kroA150	4	-0.16	0.7570	talent_10_70_8	4	-0.15	0.7838
kroA150	5	0.06	0.9120	talent_10_70_8	5	0.58	0.2245
rat195	0	-0.24	0.6440	talent_10_100_5	0	0.38	0.4516
rat195	1	0.09	0.8649	talent_10_100_5	1	0.79	0.0629
rat195	2	-0.03	0.9617	talent_10_100_5	2	0.63	0.1823
rat195	3	0.26	0.6119	talent_10_100_5	3	-0.03	0.9542
rat195	4	0.74	0.0957	talent_10_100_5	4	-0.01	0.9866
rat195	5	0.02	0.9640	talent_10_100_5	5	-0.03	0.9591
gr202	0	0.84	0.0363^{*B}	talent_10_150_4	0	0.87	0.0238^{*B}
gr202	1	-0.36	0.4851	talent_10_150_4	1	0.90	0.0148^{*B}
gr202	2	0.02	0.9700	talent_10_150_4	2	0.89	0.0177^{*B}
gr202	3	-0.31	0.5548	talent_10_150_4	3	0.25	0.6340
gr202	4	0.09	0.8589	talent_10_150_4	4	0.74	0.0939
gr202	5	0.13	0.8053	talent_10_150_4	5	0.58	0.2252
att532	0	-0.06	0.9106	talent_10_175_4	0	0.62	0.1893
att532	1	-0.76	0.0779	talent_10_175_4	1	0.94	0.0048^{*B}
att532	2	0.43	0.3951	talent_10_175_4	2	0.98	0.0006*
att532	3	0.48	0.3317	talent_10_175_4	3	0.69	0.1315
att532	4	-0.37	0.4753	talent_10_175_4	4	-0.23	0.6596
att532	5	0.43	0.4002	talent_10_175_4	5	0.23	0.6681
rat575	0	0.43	0.3909	talent_10_200_5	0	0.73	0.0973
rat575	1	0.76	0.0812	talent_10_200_5	1	0.96	0.0027^{*B}
rat575	2	0.69	0.1296	talent_10_200_5	2	0.98	0.0004*
rat575	3	0.89	0.0163^{*B}	talent_10_200_5	3	0.91	0.0108^{*B}
rat575	4	0.68	0.1362	talent_10_200_5	4	0.31	0.5472
rat575	5	0.72	0.1042	talent_10_200_5	5	0.20	0.7055

Table C.6: Pearson Product-Moment Correlations of P_{GB} comparing MMAS and GBAS varying β. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. r-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$.)

Problem	β	r	P-Value	Problem	β	r	P-Value
kroA150	0	0.45	0.3704	esc16g	0	-0.29	0.5734
kroA150	1	-0.19	0.7217	bur26d	0	-0.54	0.2724
kroA150	2	-0.44	0.3778	tai35b	0	-0.68	0.1354
kroA150	3	0.25	0.6265	tai60b	0	-0.39	0.4491
kroA150	4	-0.50	0.3169	talent_10_70_8	0	0.85	0.0338^{*B}
kroA150	5	0.70	0.1244	talent_10_70_8	1	0.49	0.3285
rat195	0	0.82	0.0453^{*B}	talent_10_70_8	2	0.77	0.0719
rat195	1	0.95	0.0041^{*B}	talent_10_70_8	3	0.67	0.1419
rat195	2	0.82	0.0451^{*B}	talent_10_70_8	4	0.77	0.0743
rat195	3	0.78	0.0702	talent_10_70_8	5	0.68	0.1401
rat195	4	0.48	0.3374	talent_10_100_5	0	0.77	0.0749
rat195	5	0.55	0.2587	talent_10_100_5	1	0.77	0.0759
gr202	0	0.61	0.1951	talent_10_100_5	2	0.93	0.0079^{*B}
gr202	1	0.52	0.2943	talent_10_100_5	3	0.57	0.2386
gr202	2	0.04	0.9417	talent_10_100_5	4	0.76	0.0806
gr202	3	-0.20	0.6996	talent_10_100_5	5	0.53	0.2838
gr202	4	0.07	0.8895	talent_10_150_4	0	0.50	0.3081
gr202	5	-0.22	0.6731	talent_10_150_4	1	0.93	0.0076^{*B}
att532	0	0.30	0.5667	talent_10_150_4	2	0.39	0.4397
att532	1	0.14	0.7981	talent_10_150_4	3	0.30	0.5660
att532	2	-0.20	0.7105	talent_10_150_4	4	0.44	0.3768
att532	3	0.31	0.5520	talent_10_150_4	5	0.42	0.4092
att532	4	-0.10	0.8543	talent_10_175_4	0	0.60	0.2034
att532	5	0.70	0.1189	talent_10_175_4	1	0.49	0.3279
rat575	0	0.45	0.3664	talent_10_175_4	2	0.65	0.1660
rat575	1	0.81	0.0508	talent_10_175_4	3	0.49	0.3276
rat575	2	0.51	0.3000	talent_10_175_4	4	0.43	0.3928
rat575	3	0.66	0.1519	talent_10_175_4	5	0.79	0.0622^{*B}
rat575	4	0.51	0.3049	talent_10_200_5	0	0.57	0.2388
rat575	5	0.80	0.0539	talent_10_200_5	1	0.41	0.4153
nug20	0	0.96	0.0027^{*}	talent_10_200_5	2	0.46	0.3563
sko100a	0	0.62	0.1913	talent_10_200_5	3	0.39	0.4468
lipa40a	0	-0.26	0.6169	talent_10_200_5	4	0.86	0.0268^{*B}
tho150	0	0.23	0.6565	talent_10_200_5	5	0.90	0.0143^{*B}

Table C.7: Pearson Product-Moment Correlations of I_{GB} comparing Ant System and GBAS varying α. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. *r*-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$, QAP is $\alpha = \frac{0.05}{8} = 0.0063$.)

Problem	α	r	P-Value	Problem	α	r	P-Value
kroA150	0	-0.56	0.2448	talent_10_70_8	0	0.50	0.3133
kroA150	1	0.92	0.0087^{*B}	talent_10_70_8	1	0.57	0.2426
kroA150	2	0.98	0.0006*	talent_10_70_8	2	0.97	0.0012*
kroA150	3	-0.21	0.6963	talent_10_70_8	3	0.99	0.0002*
kroA150	4	0.32	0.5360	talent_10_70_8	4	1.00	< 0.0001*
kroA150	5	-0.45	0.3686	talent_10_70_8	5	1.00	< 0.0001*
rat195	0	-0.69	0.1263	talent_10_100_5	0	0.21	0.6880
rat195	1	0.75	0.0849	talent_10_100_5	1	-0.56	0.2476
rat195	2	0.38	0.4580	talent_10_100_5	2	0.88	0.0199^{*B}
rat195	3	-0.71	0.1119	talent_10_100_5	3	0.98	0.0004*
rat195	4	0.08	0.8864	talent_10_100_5	4	0.99	0.0002*
rat195	5	-0.26	0.6159	talent_10_100_5	5	0.99	< 0.0001*
gr202	0	-0.49	0.3283	talent_10_150_4	0	-0.26	0.6173
gr202	1	0.71	0.1131	talent_10_150_4	1	0.08	0.8774
gr202	2	-0.38	0.4527	talent_10_150_4	2	0.80	0.0576
gr202	3	-0.60	0.2090	talent_10_150_4	3	0.88	0.0195^{*B}
gr202	4	-0.76	0.0784	talent_10_150_4	4	0.92	0.0087^{*B}
gr202	5	-0.45	0.3756	talent_10_150_4	5	0.97	0.0009*
att532	0	-0.28	0.5919	talent_10_175_4	0	0.42	0.4077
att532	1	0.88	0.0196^{*B}	talent_10_175_4	1	0.82	0.0457^{*B}
att532	2	-0.41	0.4201	talent_10_175_4	2	-0.07	0.8993
att532	3	-0.39	0.4444	talent_10_175_4	3	0.79	0.0627
att532	4	-0.15	0.7745	talent_10_175_4	4	0.89	0.0162^{*B}
att532	5	-0.54	0.2714	talent_10_175_4	5	0.92	0.0094^{*B}
rat575	0	0.49	0.3186	talent_10_200_5	0	-0.12	0.8221
rat575	1	0.68	0.1388	talent_10_200_5	1	0.79	0.0600
rat575	2	-0.19	0.7230	talent_10_200_5	2	-0.67	0.1448
rat575	3	0.73	0.0969	talent_10_200_5	3	0.78	0.0657
rat575	4	0.15	0.7697	talent_10_200_5	4	0.88	0.0207^{*B}
rat575	5	-0.08	0.8788	talent_10_200_5	5	0.87	0.0252^{*B}

Table C.8: Pearson Product-Moment Correlations of I_{GB} comparing Ant System and GBAS varying β. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. *r*-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$.)

Problem	β	r	P-Value	Problem	β	r	P-Value
kroA150	0	0.97	0.0017*	esc16g	0	-0.29	0.5734
kroA150	1	0.99	0.0001*	bur26d	0	0.94	0.0061*
kroA150	2	0.96	0.0029^{*B}	tai35b	0	0.85	0.0304^{*B}
kroA150	3	0.97	0.0016*	tai60b	0	0.97	0.0014*
kroA150	4	0.94	0.0049^{*B}	talent_10_70_8	0	0.84	0.0349^{*B}
kroA150	5	0.96	0.0027^{*B}	talent_10_70_8	1	0.85	0.0316^{*B}
rat195	0	0.98	0.0006*	talent_10_70_8	2	0.96	0.0021^{*B}
rat195	1	0.97	0.0015*	talent_10_70_8	3	0.92	0.0089^{*B}
rat195	2	0.99	0.0002*	talent_10_70_8	4	0.57	0.2352
rat195	3	0.96	0.0024^{*B}	talent_10_70_8	5	0.77	0.0716
rat195	4	0.98	0.0005*	talent_10_100_5	0	1.00	< 0.0001*
rat195	5	0.99	< 0.0001*	talent_10_100_5	1	0.97	0.0018^{*B}
gr202	0	0.96	0.0018^{*B}	talent_10_100_5	2	0.96	0.0019^{*B}
gr202	1	1.00	< 0.0001*	talent_10_100_5	3	0.77	0.0708
gr202	2	0.93	0.0074^{*B}	talent_10_100_5	4	0.66	0.1520
gr202	3	0.96	0.0020^{*B}	talent_10_100_5	5	0.70	0.1204
gr202	4	0.80	0.0562	talent_10_150_4	0	0.97	0.0009*
gr202	5	0.96	0.0026^{*B}	talent_10_150_4	1	0.99	0.0001*
att532	0	0.86	0.0270^{*B}	talent_10_150_4	2	0.81	0.0528
att532	1	0.90	0.0135^{*B}	talent_10_150_4	3	0.56	0.2524
att532	2	0.60	0.2068	talent_10_150_4	4	0.29	0.5746
att532	3	0.83	0.0431^{*B}	talent_10_150_4	5	0.37	0.4640
att532	4	0.98	0.0005*	talent_10_175_4	0	0.95	0.0033^{*B}
att532	5	0.90	0.0156^{*B}	talent_10_175_4	1	0.96	0.0024*
rat575	0	0.65	0.1603	talent_10_175_4	2	0.85	0.0323^{*B}
rat575	1	0.96	0.0028^{*B}	talent_10_175_4	3	0.52	0.2876
rat575	2	0.92	0.0105^{*B}	talent_10_175_4	4	0.33	0.5166
rat575	3	0.91	0.0123^{*B}	talent_10_175_4	5	0.61	0.1966
rat575	4	0.96	0.0019^{*B}	talent_10_200_5	0	0.97	0.0018^{*B}
rat575	5	0.98	0.0007*	talent_10_200_5	1	0.98	0.0004*
nug20	0	0.92	0.0096^{*B}	talent_10_200_5	2	0.82	0.0474^{*B}
sko100a	0	0.95	0.0033^{*}	talent_10_200_5	3	0.51	0.3001
lipa40a	0	0.83	0.0395^{*B}	talent_10_200_5	4	0.70	0.1209
tho150	0	0.99	0.0002*	talent_10_200_5	5	0.63	0.1830

Table C.9: Pearson Product-Moment Correlations of I_{GB} comparing MMAS and GBAS varying α. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. *r*-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$, QAP is $\alpha = \frac{0.05}{8} = 0.0063$.)

Problem	α	r	P-Value	Problem	α	r	P-Value
kroA150	0	0.30	0.5594	talent_10_70_8	0	0.26	0.6197
kroA150	1	0.90	0.0135^{*B}	talent_10_70_8	1	-0.65	0.1607
kroA150	2	0.38	0.4552	talent_10_70_8	2	0.93	0.0078^{*B}
kroA150	3	-0.02	0.9671	talent_10_70_8	3	0.99	< 0.0001*
kroA150	4	-0.37	0.4664	talent_10_70_8	4	0.99	0.0003*
kroA150	5	-0.33	0.5253	talent_10_70_8	5	0.99	0.0002*
rat195	0	0.28	0.5918	talent_10_100_5	0	0.12	0.8258
rat195	1	0.95	0.0044^{*B}	talent_10_100_5	1	-0.81	0.0486^{*B}
rat195	2	0.64	0.1751	talent_10_100_5	2	0.91	0.0108^{*B}
rat195	3	-0.28	0.5856	talent_10_100_5	3	0.99	< 0.0001*
rat195	4	0.11	0.8396	talent_10_100_5	4	0.97	0.0014*
rat195	5	-0.04	0.9471	talent_10_100_5	5	0.99	< 0.0001*
gr202	0	0.06	0.9097	talent_10_150_4	0	-0.33	0.5197
gr202	1	0.93	0.0077^{*B}	talent_10_150_4	1	0.28	0.5892
gr202	2	-0.14	0.7969	talent_10_150_4	2	0.74	0.0944
gr202	3	-0.35	0.4992	talent_10_150_4	3	0.98	0.0009*
gr202	4	0.01	0.9852	talent_10_150_4	4	0.97	0.0010*
gr202	5	-0.61	0.1954	talent_10_150_4	5	0.95	0.0038^{*B}
att532	0	-0.73	0.0978	talent_10_175_4	0	0.70	0.1206
att532	1	0.92	0.0096^{*B}	talent_10_175_4	1	0.87	0.0260^{*B}
att532	2	0.14	0.7876	talent_10_175_4	2	0.65	0.1587
att532	3	0.23	0.6647	talent_10_175_4	3	0.93	0.0063^{*B}
att532	4	-0.50	0.3176	talent_10_175_4	4	0.98	0.0005*
att532	5	-0.51	0.3017	talent_10_175_4	5	1.00	< 0.0001*
rat575	0	-0.15	0.7793	talent_10_200_5	0	-0.08	0.8843
rat575	1	0.13	0.8117	talent_10_200_5	1	0.94	0.0047^{*B}
rat575	2	0.04	0.9474	talent_10_200_5	2	0.65	0.1597
rat575	3	0.03	0.9552	talent_10_200_5	3	0.98	0.0004*
rat575	4	-0.57	0.2344	talent_10_200_5	4	0.98	0.0006*
rat575	5	0.16	0.7583	talent_10_200_5	5	0.98	0.0008*

Table C.10: Pearson Product-Moment Correlations of I_{GB} comparing MMAS and GBAS varying β. (NA (0/1)=Those points where all values are equal and therefore have no standard deviation. *=Significant Correlations requiring the p-value is less than 0.05. r-coefficient given to 2 decimal places, p-values given to 4 decimal places. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{30} = 0.0017$.)

Problem	N	ρ_{alg}	AS		MMAS	
			r	p-value	r	p-value
kroA150	150	0.70	0.38	0.0663	0.33	0.1124
kroA150	150	0.90	0.21	0.3174	0.53	0.0074^{*B}
kroA150	150	0.95	0.07	0.7343	-0.11	0.5955
kroA150	150	0.99	NA (1)	NA (0)*	NA (1)	NA (0)*
rat195	195	0.70	0.47	0.0206^{*B}	0.56	0.0044^{*B}
rat195	195	0.90	0.29	0.1743	0.48	0.0185^{*B}
rat195	195	0.95	0.43	0.0346^{*B}	0.51	0.0102^{*B}
rat195	195	0.99	0.13	0.5516	0.28	0.1860
gr202	202	0.70	0.43	0.0357^{*B}	0.61	0.0014*
gr202	202	0.90	0.07	0.7407	0.24	0.2638
gr202	202	0.95	0.17	0.4406	0.29	0.1689
gr202	202	0.99	0.08	0.7180	0.14	0.5085
att532	532	0.70	0.38	0.0656	0.55	0.0059^{*B}
att532	532	0.90	0.61	0.0014*	0.76	< 0.0001*
att532	532	0.95	0.89	< 0.0001*	0.90	< 0.0001*
att532	532	0.99	0.94	< 0.0001*	0.91	< 0.0001*
rat575	575	0.70	0.62	0.0013*	0.72	< 0.0001*
rat575	575	0.90	0.73	< 0.0001*	0.85	< 0.0001*
rat575	575	0.95	0.94	< 0.0001*	0.95	< 0.0001*
rat575	575	0.99	0.95	< 0.0001*	0.94	< 0.0001*
nug20	20	0.70	-0.08	0.7265	-0.22	0.3604
nug20	20	0.90	0.20	0.3960	-0.07	0.7710
nug20	20	0.95	-0.02	0.9202	0.06	0.8077
nug20	20	0.99	0.11	0.6346	0.18	0.4574
sko100a	100	0.70	-0.29	0.1724	-0.16	0.4427
sko100a	100	0.90	-0.07	0.7401	0.03	0.8906
sko100a	100	0.95	0.12	0.5892	0.44	0.0310^{*B}
sko100a	100	0.99	0.08	0.7273	-0.01	0.9518
lipa40a	40	0.70	-0.00	0.9926	-0.06	0.7324
lipa40a	40	0.90	0.09	0.5944	0.02	0.9170
lipa40a	40	0.95	-0.02	0.9008	-0.11	0.4845
lipa40a	40	0.99	-0.21	0.2003	0.20	0.2115
tho150	150	0.70	-0.42	0.0434^{*B}	0.09	0.6846
tho150	150	0.90	-0.26	0.2278	0.34	0.1041
tho150	150	0.95	0.42	0.0404^{*B}	0.74	< 0.0001*
tho150	150	0.99	0.07	0.7336	0.22	0.2948

Table C.11: Correlations between AS,MMAS and GBAS, and their associated p-values showing their significance for P_{GB}. (*=Significant correlations given to 4 decimal places, *r* values given to 2 decimal places. NA (0/1)=Those points where all values are equal and therefore have no standard deviation. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{20} = 0.0025$, QAP is $\alpha = \frac{0.05}{32} = 0.0016$.)

Problem	N	ρ_{alg}	AS		MMAS	
			r	p-value	r	p-value
esc16g	16	0.70	NA (1)	NA (0)*	NA (1)	NA (0)*
esc16g	16	0.90	NA (1)	NA (0)*	NA (1)	NA (0)*
esc16g	16	0.95	NA (1)	NA (0)*	NA (1)	NA (0)*
esc16g	16	0.99	NA (1)	NA (0)*	NA (1)	NA (0)*
bur26d	26	0.70	-0.02	0.9416	-0.37	0.0607
bur26d	26	0.90	-0.13	0.5299	0.13	0.5293
bur26d	26	0.95	0.26	0.1963	0.10	0.6100
bur26d	26	0.99	0.01	0.9772	-0.04	0.8315
tai35b	35	0.70	-0.17	0.3343	0.17	0.3237
tai35b	35	0.90	-0.25	0.1552	-0.06	0.7189
tai35b	35	0.95	0.34	0.0462^{*B}	0.08	0.6667
tai35b	35	0.99	-0.12	0.4902	0.43	0.0100^{*B}
tai60b	60	0.70	0.20	0.3514	-0.03	0.8719
tai60b	60	0.90	-0.22	0.2933	0.11	0.5995
tai60b	60	0.95	-0.00	0.9962	0.08	0.7038
tai60b	24	0.99	0.13	0.5380	0.54	0.0066^{*B}
talent_10_70_8	70	0.70	0.02	0.9425	0.05	0.8221
talent_10_70_8	70	0.90	0.23	0.2809	0.17	0.4168
talent_10_70_8	70	0.95	-0.26	0.2250	-0.14	0.5078
talent_10_70_8	70	0.99	0.37	0.0776	0.41	0.04831^{*B}
talent_10_100_5	100	0.70	0.10	0.6454	0.08	0.7132
talent_10_100_5	100	0.90	-0.22	0.2964	-0.58	0.0030^{*B}
talent_10_100_5	100	0.95	0.34	0.1058	0.27	0.2021
talent_10_100_5	100	0.99	-0.02	0.9131	0.26	0.2114
talent_10_150_4	100	0.70	-0.18	0.3955	0.51	0.0110^{*B}
talent_10_150_4	150	0.90	0.34	0.0987	0.02	0.9305
talent_10_150_4	150	0.95	0.48	0.0175^{*B}	0.19	0.3849
talent_10_150_4	150	0.99	0.47	0.0209^{*B}	0.44	0.0325^{*B}
talent_10_175_4	175	0.70	-0.44	0.0312^{*B}	-0.21	0.3217
talent_10_175_4	175	0.90	0.24	0.2527	0.40	0.0531
talent_10_175_4	175	0.95	0.88	< 0.0001*	0.88	< 0.0001*
talent_10_175_4	175	0.99	0.24	0.2487	0.26	0.2272
talent_10_200_5	200	0.70	0.11	0.5978	-0.02	0.9087
talent_10_200_5	200	0.90	0.89	< 0.0001*	0.89	< 0.0001*
talent_10_200_5	200	0.95	0.87	< 0.0001*	0.97	< 0.0001*
talent_10_200_5	200	0.99	0.68	0.0002*	0.59	0.0025*

Table C.12: *Continued.* Correlations between AS,MMAS and GBAS, and their associated significance p-values for P_{GB}. (*=Significant correlations given to 4 decimal places, *r* values given to 2 decimal places. NA (0/1)=Those points where all values are equal and therefore have no standard deviation. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{20} = 0.0025$, QAP is $\alpha = \frac{0.05}{32} = 0.0016$.)

Problem	N	ρ_{alg}	AS		MMAS	
			r	p-value	r	p-value
kroA150	150	0.70	0.28	0.1796	0.53	$0.0075^{\star B}$
kroA150	150	0.90	0.77	< 0.0001*	0.74	< 0.0001*
kroA150	150	0.95	0.33	0.1182	0.84	< 0.0001*
kroA150	150	0.99	0.09	0.6635	-0.36	0.0859
rat195	195	0.70	0.75	< 0.0001*	0.71	< 0.0001*
rat195	195	0.90	0.88	< 0.0001*	0.87	< 0.0001*
rat195	195	0.95	0.93	< 0.0001*	0.86	< 0.0001*
rat195	195	0.99	0.31	0.1357	-0.24	0.2598
gr202	202	0.70	0.82	< 0.0001*	0.85	< 0.0001*
gr202	202	0.90	0.53	$0.0081^{\star B}$	0.54	$0.0059^{\star B}$
gr202	202	0.95	0.73	< 0.0001*	0.88	< 0.0001*
gr202	202	0.99	0.39	0.0573	0.04	0.8680
att532	532	0.70	0.48	$0.0172^{\star B}$	0.85	< 0.0001*
att532	532	0.90	0.76	< 0.0001*	0.92	< 0.0001*
att532	532	0.95	0.87	< 0.0001*	0.96	< 0.0001*
att532	532	0.99	0.89	< 0.0001*	0.88	< 0.0001*
rat575	575	0.70	0.59	0.0023*	0.79	< 0.0001*
rat575	575	0.90	0.76	< 0.0001*	0.92	< 0.0001*
rat575	575	0.95	0.89	< 0.0001*	0.95	< 0.0001*
rat575	575	0.99	0.86	< 0.0001*	0.93	< 0.0001*
nug20	20	0.70	0.00	1.000	0.44	0.0506
nug20	20	0.90	0.54	$0.0140^{\star B}$	0.46	$0.0413^{\star B}$
nug20	20	0.95	0.46	$0.0437^{\star B}$	0.27	0.2563
nug20	20	0.99	0.23	0.3299	-0.03	0.8911
sko100a	100	0.70	0.30	0.1583	-0.03	0.8780
sko100a	100	0.90	0.33	0.1192	0.48	$0.0186^{\star B}$
sko100a	100	0.95	-0.11	0.6084	0.03	0.8957
sko100a	100	0.99	-0.20	0.3452	0.27	0.1956
lipa40a	40	0.70	0.53	0.0004*	0.05	0.7713
lipa40a	40	0.90	0.61	< 0.0001*	-0.05	0.7794
lipa40a	40	0.95	0.35	$0.0265^{\star B}$	0.06	0.7314
lipa40a	40	0.99	0.12	0.4455	-0.05	0.7588
tho150	150	0.70	0.55	$0.0052^{\star B}$	0.66	0.0005*
tho150	150	0.90	0.38	0.0660	0.41	$0.0490^{\star B}$
tho150	150	0.95	0.93	< 0.0001*	0.95	< 0.0001*
tho150	150	0.99	0.52	$0.0095^{\star B}$	0.45	$0.0293^{\star B}$

Table C.13: Correlations between AS/MMAS and GBAS, and their associated significance p-values for I_{GB}. (*=Significant correlations given to 4 decimal places, *r* values given to 2 decimal places. NA (0/1)=Those points where all values are equal and therefore have no standard deviation. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{20} = 0.0025$, QAP is $\alpha = \frac{0.05}{32} = 0.0016$.)

Problem	N	ρ_{alg}	AS		MMAS	
			r	p-value	r	p-value
esc16g	16	0.70	-0.20	0.4493	-0.33	0.2071
esc16g	16	0.90	-0.29	0.2797	-0.26	0.3243
esc16g	16	0.95	-0.35	0.1811	0.47	0.0694
esc16g	16	0.99	0.05	0.8401	0.40	0.1255
bur26d	26	0.70	0.46	0.0183^{*B}	0.54	0.0047^{*B}
bur26d	26	0.90	0.33	0.0998	0.08	0.7046
bur26d	26	0.95	0.36	0.0749	0.44	0.0229^{*B}
bur26d	26	0.99	-0.08	0.7149	-0.12	0.5440
tai35b	35	0.70	0.58	0.0003*	0.20	0.2475
tai35b	35	0.90	0.57	0.0003*	0.38	0.0260^{*B}
tai35b	35	0.95	-0.17	0.3149	0.64	< 0.0001*
tai35b	35	0.99	-0.28	0.1021	-0.04	0.7991
tai60b	60	0.70	0.62	0.0012*	0.21	0.3323
tai60b	60	0.90	0.72	< 0.0001*	0.39	0.0596
tai60b	60	0.95	-0.59	0.0024^{*B}	0.35	0.0894
tai60b	60	0.99	-0.15	0.4782	0.37	0.0736
talent_10_70_8	70	0.70	0.84	< 0.0001*	-0.02	0.9418
talent_10_70_8	70	0.90	0.83	< 0.0001*	0.57	0.0033^{*B}
talent_10_70_8	70	0.95	0.77	< 0.0001*	0.44	0.0329^{*B}
talent_10_70_8	70	0.99	-0.11	0.6150	0.58	0.0030^{*B}
talent_10_100_5	100	0.70	0.89	< 0.0001*	0.22	0.3054
talent_10_100_5	100	0.90	0.83	< 0.0001*	0.55	0.0053^{*B}
talent_10_100_5	100	0.95	0.44	0.0294^{*B}	0.64	0.0008*
talent_10_100_5	100	0.99	0.10	0.6580	0.19	0.3846
talent_10_150_4	150	0.70	0.88	< 0.0001*	0.32	0.1324
talent_10_150_4	150	0.90	0.80	< 0.0001*	0.77	< 0.0001*
talent_10_150_4	150	0.95	0.53	0.0083^{*B}	0.48	0.0182^{*B}
talent_10_150_4	150	0.99	0.26	0.2118	0.45	0.0271^{*B}
talent_10_175_4	175	0.70	0.90	< 0.0001*	0.29	0.1673
talent_10_175_4	175	0.90	0.82	< 0.0001*	0.86	< 0.0001*
talent_10_175_4	175	0.95	1.00	0.0000*	0.99	0.0000*
talent_10_175_4	175	0.99	0.54	0.0065^{*B}	0.64	0.0008*
talent_10_200_5	200	0.70	0.92	< 0.0001*	0.54	0.0065^{*B}
talent_10_200_5	200	0.90	0.91	< 0.0001*	0.92	< 0.0001*
talent_10_200_5	200	0.95	0.99	0.0000*	1.00	0.0000*
talent_10_200_5	200	0.99	0.78	< 0.0001*	0.77	< 0.0001*

Table C.14: *Continued.* Correlations between AS/MMAS and GBAS, and their associated significance p-values for I_{GB}. (*=Significant correlations given to 4 decimal places, *r* values given to 2 decimal places. NA (0/1)=Those points where all values are equal and therefore have no standard deviation. A superscript 'B' indicates that the value is not significant when the alpha undergoes the Bonferroni Correction. The Bonferroni corrected αs are as follows: TSP,TS is $\alpha = \frac{0.05}{20} = 0.0025$, QAP is $\alpha = \frac{0.05}{32} = 0.0016$.)

Bibliography

Aarts, E. and Lenstra, J. K., editors (1997). *Local Search in combinatorial optimization*. Princeton University Press.

Acan, A. (2004). An external memory implementation in ant colony optimization. In Dorigo et al. [2004], pages 73–82.

Adelson, R. M., Norman, J. M., and Laporte, G. (1976). A dynamic programming formulationa with diverse applications. *Operational Research Quarterly*, 27:119–121.

Anstreicher, K. M., Brixius, N. W., Linderoth, J., and Goux, J. P. (2002). Solving large quadratic assignment problems on computational grids. In *Mathematical Programming*, volume 91 of *B*, pages 563–588.

Appleby, S. and Steward, S. (1994). Mobile software agents for control in telecommunications networks. *British Telecom Technology Journal*, 12(2):104–113.

Armour, G. C. and Buffa, E. S. (1963). Heuristic algorithm and simulation approach to relative location of facilities. *Management Science*, 9:294–309.

Barr, R. S., Golden, B. L., Kelly, J. P., Rescende, M., and Stewart, W. (1995). Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1(1):9–32.

Bautista, J. and Pereira, J. (2002). Ant algorithms for assembly line balancing. In Dorigo et al. [2002a], pages 65–75.

Bianchi, L., Gambardella, L. M., and Dorigo, M. (2002a). Ant colony optimization for the probabilistic traveling salesman problem. In *Abstracts of ECCO XV -Conference of the European Chapter on Combinatorial Optimisation*, Lugano, Switzerland.

Bianchi, L., Gambardella, L. M., and Dorigo, M. (2002b). Solving the homogeneous

probabilistic traveling salesman problem by the aco metaheuristic. In Dorigo et al. [2002a], pages 176–187.

Birattari, M., Di Caro, G., and Dorigo, M. (2002a). Toward the formal foundation of ant programming. In Dorigo et al. [2002a], pages 188–201.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002b). A racing algorithm for configuring metaheuristics. In et al., W. B. L., editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18. Morgan Kaufmann, San Francisco, CA, USA.

Blum, C. (2002a). Aco applied to group shop scheduling: A case study on intensification and diversification. In Dorigo et al. [2002a], pages 14–27.

Blum, C. (2002b). Ant colony optimization for the edge-weighted k-cardinality tree problem. In *Proceedings of the 2002 Genetic and Evolutionary Computation Conference, GECCO'02*, pages 27–34.

Blum, C. (2005). Beam-aco–hybridizing ant colony optimization with beam search: an application to open shop scheduling. In *Computers and Operations Research*, volume 32,6, pages 1565–1591.

Blum, C. and Dorigo, M. (2004). Deception in ant colony optimization. In Dorigo et al. [2004], pages 118–129.

Blum, C., Roli, A., and Dorigo, M. (2001). HC–ACO: The hyper-cube framework for Ant Colony Optimization. In *Proceedings of MIC'2001 – Meta–heuristics International Conference*, volume 2, pages 399–403, Porto, Portugal. Also available as technical report TR/IRIDIA/2001-16, IRIDIA, Université Libre de Bruxelles.

Blum, C. and Sampels, M. (2002a). Ant colony optimization for fop shop scheduling: A case study on di erent pheromone representations. Technical report, IRIDIA. TR/IRIDIA/2002-3 February 2002.

Blum, C. and Sampels, M. (2002b). When model bias is stronger than selection pressure. In *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature, PPSN'02*. Also available as technical report TR/IRIDIA/2002-06, IRIDIA, Université Libre de Bruxelles.

Blum, C., Sampels, M., and Zlochin, M. (2002). On a particularity in model-based search. In *GECCO*, pages 35–42.

Boffey, T. B. (1982). *Graph Theory in Operations Research*. The MacMillan Press Ltd.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (2000). Inspiration for optimization from social insect behaviour. *Nature*, 406:39–42.

Bonabeau, E., Henaux, F., Guérin, S., Snyers, D., Kuntz, P., and Theraulaz, G. (1998). Routing in telecommunications networks with "smart" ant-like agents. In *Intelligent Agents for Telecommunications Applications '98 (IATA'98)*.

Botee, H. M. and Bonabeau, E. (1998). Evolving ant colony optimization. *Advanced Complex Systems*, 1:149–159.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Bullnheimer, B., Hartl, R., and Strauss, C. (1997a). Applying the ant system to the vehicle routing problem. Bullnheimer B., R.F. Hartl and C. Strauss (1997), Applying the Ant System to the Vehicle Routing Problem, Presented at the 2nd Metaheuristic International Conference, Sophia-Antipolis, France.

Bullnheimer, B., Hartl, R., and Strauss, C. (1997b). An improved ant system algorithm for the vehicle routing problem. B. Bullnheimer, R.F. Hartl, and C. Strauss. An Improved Ant System Algorithm for the Vehicle Routing Problem. Annals of Operations Research, 89:319-328, 1999.

Bullnheimer, B., Hartl, R., and Strauss, C. (1997c). A new rank based version of the ant system — a computational study. B. Bullnheimer, Richard F. Hartl, and C. Strauss. A New Rank Based Version of the Ant System — A Computational Study. Technical report, University of Viena, Institute of Management Science, 1997.

Bullnheimer, B., Kotsis, G., and Strauss, C. (1997d). Parallelization strategies for the Ant System. B. Bullnheimer, G. Kotsis, and C. Strauss. Parallelization Strategies for the Ant System. Technical Report POM 9/97, University of Vienna, 1997.

Burkard, R. E., Çela, E., Karisch, S. E., and Rendl, F. (1996). QAPLIB. http://www.opt.math.tu-graz.ac.at/qaplib/.

Carlson, R. C. and Nemhauser, G. L. (1966). Scheduling to minimise cost. *Operations Research*, 14:52–58.

Cheng, T. C. E., Diamond, J., and Lin, B. M. T. (1993). Optimal scheduling in film

production to minimize talent hold cost. *Journal of Optimization Theory and Applications*, 79(3):197–206.

Cohen, P. R. (1995). *Empirical methods for artificial intelligence*. MIT Press, Cambridge, MA, USA.

Colorni, A., Dorigo, M., and Maniezzo, V. (1991). Distributed Optimization by Ant Colonies. In F.Varela and P.Bourgine, editors, *Proceedings of the First European Conference on Artificial Life*, pages 134–142. Elsevier Publishing.

Colorni, A., Dorigo, M., and Maniezzo, V. (1992). An Investigation of Some Properties of an Ant Algorithm. In R.Männer and B.Manderick, editors, *Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92)*, pages 509–520. Elsevier Publishing.

Coolican, H. (1994). *Research Methods and Statistics in Psychology*. Hodder and Stoughton.

Cordon, O., Casillas, J., and Herrera, F. (2000a). Learning Fuzzy Rules Using Ant Colony Optimization. In *Proc. ANTS'2000 – From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms*, pages 13–21.

Cordón, O., de Viana, I. F., and Herrera, F. (2002). Analysis of the best-worst ant system and its variants on the qap. In Dorigo et al. [2002a], pages 228–234.

Cordon, O., de Viana, I. F., Herrera, F., and Moreno, L. (2000b). A new aco model integrating evolutionary computation concepts: The best-worst ant system. In Dorigo, M., Middendorf, M., and Stützle, T., editors, *Abstract Proceedings of ANTS2000-From Ant Colonies to Artificial Ants: A Series of International Workshops on Ant Algorithms*, pages 22–29, Université Libre de Bruxelles, Belgium.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press and McGraw-Hill, second edition.

Costa, D. and Hertz, A. (1997). Ants can colour graphs. *Journal of the Operational Research Society*, 48:295–305.

Di Caro, G. and Dorigo, M. (1997). AntNet: A mobile agents approach to adaptive routing. Technical Report 12, Universite Libre de Bruxelles, IRIDIA.

Di Caro, G. and Dorigo, M. (1998a). An adaptive multi-agent routing algorithm in-

spired by ants behavior. In *Proceedings of PART98 — 5th Annual Australasian Conference on Parallel and Real-Time Systems*, pages 261–272. Springer-Verlag.

Di Caro, G. and Dorigo, M. (1998b). Ant colonies for adaptive routing in packet-switched communications networks. *Lecture Notes in Computer Science*, 1498:673–682.

Di Caro, G. and Dorigo, M. (1998c). AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9:317–365.

Di Caro, G. and Dorigo, M. (1998d). Mobile agents for adaptive routing. In *31st Hawaii International Conference on System Science*, Big Island of Hawaii.

Di Caro, G. and Dorigo, M. (1998e). Two ant colony algorithms for best-effort routing in datagram networks. G. Di Caro and M. Dorigo. Two ant colony algorithms for best-effort routing in datagram networks. In Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98), pages 541–546. IASTED/ACTA Press, 1998.

Donati, A., Gambardella, L., Casagrande, N., Rizzoli, A., and Montemanni, R. (2003). Time dependent vehicle routing problem with an ant colony system. IDSIA report IDSIA-02-03, 20-Jan-03.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy. in Italian.

Dorigo, M., Birattari, M., Blum, C., Gambardella, L. M., Mondada, F., and Stützle, T., editors (2004). *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5 - 8, 2004, Proceedings*, volume 3172 of *Lecture Notes in Computer Science*. Springer.

Dorigo, M. and Di Caro, G. (1999a). Ant Colony Optimization: A new meta-heuristic. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1470–1477, Mayflower Hotel, Washington D.C., USA. IEEE Press.

Dorigo, M. and Di Caro, G. (1999b). The Ant Colony Optimization Meta-Heuristic. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, London.

Dorigo, M., Di Caro, G., and Sampels, M., editors (2002a). *Ant Algorithms, Third*

International Workshop, ANTS 2002, Brussels, Belgium, September 12-14, 2002, Proceedings, volume 2463 of *Lecture Notes in Computer Science*. Springer.

Dorigo, M. and Gambardella, L. M. (1996). A Study of some Properties of Ant-Q. In Voigt, H., Ebeling, W., Rechenberg, I., and Schwefel, H., editors, *Proceedings of PPSN96 International Conference on Parallel Problem Solving from Nature*, pages 656–665. Springer-Verlag.

Dorigo, M. and Gambardella, L. M. (1997). Ant Colony System: A cooperative learning approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

Dorigo, M., Maniezzo, V., and Colorni, A. (1991a). The Ant System: An Autocatalytic Optimizing Process. Technical report, Politecnico di Milano, Italy. No.91-016 Revised.

Dorigo, M., Maniezzo, V., and Colorni, A. (1991b). Positive feedback as a search strategy. M. Dorigo, V. Maniezzo, and A. Colorni, Positive feedback as a search strategy, Technical Report 91016, Dipartimento di Elettronica e Informatica, Politecnico di Milano, IT, 1991.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41.

Dorigo, M., Zlochin, M., Meuleau, N., and Birattari, M. (2002b). Updating ACO pheromones using stochastic gradient ascent and cross-entropy methods. In Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., and Raidl, G., editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim*, volume 2279, pages 21–30, Kinsale, Ireland. Springer-Verlag.

Ducatelle, F. and Levine, J. (2004). Ant colony optimisation and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society*, 55(7):705–716.

Ellabib, I., Basir, O. A., and Calamai, P. (2002). An experimental study of a simple ant colony system for the vehicle routing problem with time windows. In Dorigo et al. [2002a], pages 53–64.

Elshafei, A. N. (1977). Hospital layout as a quadratic assignment problem. *Operations Research Quarterly*, 28:167–179.

Eyckelhof, C. J. and Snoek, M. (2002). Ant systems for a dynamic tsp. In Dorigo et al. [2002a], pages 88–99.

Fink, A. and Voß, S. (1999). Applications of modern heuristic search methods to pattern sequencing problems. *Computers and Operations Research*, 26:17–34.

Gambardella, L. M. and Dorigo, M. (1995). Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem. In *International Conference on Machine Learning*, pages 252–260.

Gambardella, L. M. and Dorigo, M. (1996). Solving Symmetric and Asymmetric TSPs by Ant Colonies. In *ICEC96, Proceedings of the IEEE Conference on Evolutionary Computation*, Japan.

Gambardella, L. M. and Dorigo, M. (1997). HAS-SOP: Hybrid ant system for the sequential ordering problem. Technical Report IDSIA-11-97, IDSIA.

Gambardella, L. M., É.Taillard, and Dorigo, M. (1997). Ant Colonies for the QAP. Technical Report 4-97, IDSIA.

Gambardella, L. M., Taillard, É., and Agazzi, G. (1999a). Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization*, pages 63–76, London, UK. McGraw-Hill.

Gambardella, L. M., Taillard, É., and M., D. (1999b). Ant colonies for the Quadratic Assignment Problem. *Journal of the Operational Research Society*, 50:167–176.

Gandibleux, X., Delorme, X., and T'Kindt, V. (2004). An ant colony optimisation algorithm for the set packing problem. In Dorigo et al. [2004], pages 49–60.

Gentleman, R. and Ihaka, R. (1997). The r project for statistical computing. http://www.r-project.org/.

Geoffrion, A. M. and Graves, G. W. (1976). Scheduling parallel production lines with changeover costs: Practical applications of a quadratic assignment/lp approach. *Operations Research*, 24:595–610.

Gilmore, P. C. (1962). Optimal and suboptimal algorithms for the quadratic assignment

problem. *Journal of the Society for Industrial and Applied Mathematics*, 10(2):305–313.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, University of Colorado, Boulder.

Gómez, O. and Barán, B. (2004). Reasons of aco's success in tsp. In Dorigo et al. [2004], pages 226–237.

Goss, S., Beckers, R., Deneubourg, J. L., Aron, S., and Pasteels, J. M. (1990). How trail laying and trail following can solve foraging problems for ant colonies. In Huges, R. N., editor, *Behavioural Mechanisms of Food Selection*, volume 20G, Berlin. Springer-Verlag.

Gregory, P., Miller, A., and Prosser, P. (2004). Solving the rehearsal problem with planning and model checking. In *Proceedings of ECAI'04 Workshop on Modelling and Solving Problems with Constraints*.

Guntsch, M. and Middendorf, M. (2001). Pheromone modification strategies for ant algorithms applied to dynamic TSP. In Boers, E. J. W., Cagnoni, S., Gottlieb, J., Hart, E., Lanzi, P. L., Raidl, G., Smith, R. E., and Tijink, H., editors, *Applications of Evolutionary Computing. EvoWorkshops2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM. Proceedings*, volume 2037, pages 213–222, Como, Italy. Springer-Verlag.

Guntsch, M. and Middendorf, M. (2002a). Applying Population Based ACO to Dynamic Optimization Problems. In Dorigo et al. [2002a], pages 111–122.

Guntsch, M. and Middendorf, M. (2002b). A population based approach for ACO. In Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., and Raidl, G., editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim*, volume 2279, pages 71–80, Kinsale, Ireland. Springer-Verlag.

Guntsch, M., Middendorf, M., and Schmeck, H. (2001). An Ant Colony Optimization approach to dynamic TSP. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 860–867, San Francisco, California, USA. Morgan Kaufmann.

Gurney, K. (1997). *An Introduction to Neural Networks*, chapter 5: The Delta Rule, pages 53–56. UCL Press.

Gutjahr, W. J. (1999). A generalized convergence result for the graph-based ant system metaheuristic. Technical Report 99-09, Department of Statistics and Decision Support Systems, University of Vienna, Austria.

Gutjahr, W. J. (2000). A Graph-based Ant System and its convergence. *Future Gener. Comput. Syst.*, 16(9):873–888.

Gutjahr, W. J. (2002). Aco algorithms with guaranteed convergence to the optimal solution. *Inf. Process. Lett.*, 82(3):145–153.

Gutjahr, W. J. (2003a). A converging aco algorithm for stochastic combinatorial optimization. In Albrecht, A. A. and Steinhöfel, K., editors, *SAGA*, volume 2827 of *Lecture Notes in Computer Science*, pages 10–25. Springer.

Gutjahr, W. J. (2003b). A generalized convergence result for the graph-based ant system metaheuristic. *Probability in the Engineering and Informational Sciences*, 17:545–569.

Gutjahr, W. J. (2004). S-ACO: An Ant-Based Approach to Combinatorial Optimization Under Uncertainty. In Dorigo et al. [2004], pages 238–249.

Hadley, S., Rendl, F., and Wolkowicz, H. (1990). Bounds for the quadratic assignment problems using continuous optimization techniques. In *Proceedings of the 1st Integer Programming and Combinatorial Optimization Conference*, pages 237–248. University of Waterloo Press.

Hadley, S., Rendl, F., and Wolkowicz, H. (1992). A new lower bound via projection for the quadratic assignment problem. In *Mathematics of Operations Research*, volume 17,3, pages 727–739.

Helsgaun, K. (1998). An effective implementation if the Lin-Kernighan traveling salesman problem. Technical Report 81, Department of Computer Science, Roskilde University.

Hooker, J. N. (1996). Testing heuristics: We have it all wrong. *Journal of Heuristics*, 10732(1):33–42.

Johnson, N. L. (1977). *Urn Models and their application: an approach to modern discrete probability theory*. Wiley, New York.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Keinprasit, R. and Chongstitvatana, P. (2004). High-level synthesis by dynamic ant. *International Journal of Intelligent Systems*, 19(1-2):25–38.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, New Series*, 220(4598):671–680.

Koopmans, T. C. and Beckman, M. (1957). Assignment problems and the location of economic activities. *Econometric*, 25:53–76.

Larrañago, P. and Lozano, J. A., editors (2002). *Estimation of Distribution Algorithms*. Kluwer Academic Publishers.

Lawler, E. (1963). The quadratic assignment problem. *Management Science*, 9:586–599.

Levine, J. and Ducatelle, F. (2004). Ant colony optimisation and local search for bin packing and cutting stock problems. In *Journal of the Operational Research Society*, volume 55, number 7, pages 705–716.

Li, Y., Pardalos, P., Ramakrishnan, K., and Resende, M. (1994a). Lower bounds for the quadratic assignment problem. In *Annals of Operations Research*, volume 50, pages 387–411.

Li, Y., Pardalos, P., and Resende, M. (1994b). A greedy randomized adaptive search procedure for the quadratic assignment problem. In Pardalos, P. and Wolkowicz, H., editors, *Quadratic assignment and related problems*, volume 16 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 237–261.

Maniezzo, V. (1998). Exact and Approximate Nondeterministic Tree-search Procedures for the Quadratic Assignment Problem. Maniezzo V. (1998). Exact and Approximate Nondeterministic Tree-search Procedures for the Quadratic Assignment Problem, Report CSR 98-1, Computer Science, Univ. of Bologna.

Maniezzo, V., Boschetti, M., and Jelasity, M. (2004). An ant approach to membership overlay design. In Dorigo et al. [2004], pages 37–48.

Maniezzo, V., Carbonaro, A., Golfarelli, M., and Rizzi, S. (2001). Ants for data warehouse logical design. V. Maniezzo, A. Carbonaro, M. Golfarelli, and S. Rizzi. ANTS

for Data Warehouse Logical Design. In Proc. 4th Metaheuristics It. Cof., pages 249 254, Porto, 2001.

Maniezzo, V. and Colorni, A. (1999a). The ant system applied to the quadratic assignment problem. *IEEE Transactions on Knowledge and Data Engineering*, 11(5):769–778.

Maniezzo, V. and Colorni, A. (1999b). The Ant System applied to the Quadratic Assignment Problem. *Knowledge and Data Engineering*, 11(5):769–778.

Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. In *ACM Trans. on Modeling and Computer Simulation*, volume 8:1, pages 3–30.

McCallum, T. and Levine, J. (2002). Ant colony optimisation for generic planning domains. Dissertation, School of Informatics, University of Edinburgh.

Merkle, D. and Middendorf, M. (2000). An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In *EvoWorkshops 2000*, number 1803 in LNCS, pages 287–296. Springer Verlag.

Merkle, D. and Middendorf, M. (2001). A new approach to solve permutation scheduling problems with ant colony optimization. In *Second European Workshop on Scheduling and Timetabling (EvoStim2001)*, number 2037 in LNCS, pages 484–493. Springer Verlag.

Merkle, D. and Middendorf, M. (2002a). Ant colony optimization with the relative pheromone evaluation method. In *3rd European Workshop on Scheduling and Timetabling and 3rd European Workshop on Evolutionary Methods for AI Planning*, number 2279 in LNCS, pages 325–333. Springer Verlag.

Merkle, D. and Middendorf, M. (2002b). Modelling aco: Composed permutation problems. In Dorigo et al. [2002a], pages 149–162.

Merkle, D. and Middendorf, M. (2004). Competition controlled pheromone update for ant colony optimization. In Dorigo et al. [2004], pages 95–105.

Merkle, D., Middendorf, M., and Schmeck, H. (2000a). Ant colony optimization for resource-constrained project scheduling. In Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., and Beyer, H. G., editors, *Proceedings of the Genetic*

and Evolutionary Computation Conference (GECCO 2000), pages 893–900, San Francisco, CA, USA. Morgan Kaufmann.

Merkle, D., Middendorf, M., and Schmeck, H. (2000b). Pheromone evaluation in ant colony optimization. In *26th Annual Conference of the IEEE Electronics Society IECON-2000*, pages 2726–2731, Piscataway, USA. IEEE Press.

Meyer, B. and Ernst, A. (2004). Integrating aco and constraint propagation. In Dorigo et al. [2004], pages 166–177.

Michalewicz, Z. (1999). *Genetic Algorithms + Data Structures = Evolution Programs*, chapter 1, pages 13–18. Springer-Verlag.

Michel, R. and Middendorf, M. (1998). An island model based ant system with lookahead for the shortest supersequence problem. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 692–701. Springer-Verlag.

Michel, R. and Middendorf, M. (1999). An ACO algorithm for the shortest common supersequence problem. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization*, pages 51–61. McGraw-Hill, London.

Middendorf, M., Reischle, F., and Schmeck, H. (2000). Information exchange in multi colony ant algorithms. *Lecture Notes in Computer Science*, 1800:645+.

Monmarché, N., Slimane, M., and Venturini, G. (1999). On how the ants *Pachycondyla apicalis* are suggesting a new search algorithm. Rapport interne 214, Laboratoire d'Informatique de l'Université de Tours, E3i Tours. 17 pages.

Monmarché, N., Venturini, G., and Slimane, M. (2000). On how *Pachycondyla apicalis* ants suggest a new search algorithm. *Future Generation Computer Systems*, 16(8):937–946.

Montemanni, R., Gambardella, L., Rizzoli, A., and Donati, A. (2003). A new algorithm for a dynamic vehicle routing problem based on ant colony system. In *ODYSSEUS 2003: Second International Workshop on Freight Transportation and Logistics*, Palermo, Italy.

Montgomery, J. and Randall, M. (2002). Anti-pheromone as a tool for better exploration of search space. In Dorigo et al. [2002a], pages 100–110.

Montgomery, J., Randall, M., and Hendtlass, T. (2004). Search bias in constructive

metaheuristics and implications for ant colony optimisation. In Dorigo et al. [2004], pages 390–397.

Montresor, A. (2001). Anthill: a framework for the design and analysis of peer-to-peer systems. In *4th European Research Seminar on Advances in Distributed Systems*, Bertinoro, Italy.

Nilsson, C. (2003). Heuristics for the Traveling Salesman Problem. Technical report, Linköping University.

Nowé, A., Verbeeck, K., and Vrancx, P. (2004). Multi-type ant colony: The edge disjoint paths problem. In Dorigo et al. [2004], pages 202–213.

PARDALOS, P., RENDL, F., and WOLKOWICZ, H. (1994). The Quadratic Assignment Problem: a survey and recent developments. In Pardalos, P. and Wolkowicz, H., editors, *Quadratic assignment and related problems (New Brunswick, NJ, 1993)*, pages 1–42. Amer. Math. Soc., Providence, RI.

Pardalos, P. M., Rendl, F., and Wolkowicz, H. (1994). The Quadratic Assignment Problem: A survey and recent developments. In Pardalos, P. M. and Wolkowicz, H., editors, *Quadratic Assigment and Related Problems*, volume 16 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–42. AMS.

Park, S. and Miller, K. (1988). Random number generators: Good ones are hard to find. In *Communications of the ACM*, volume 31, issue 10, pages 1192–1201.

Pourtakdoust, S. H. and Nobahari, H. (2004). An extension of ant colony system to continuous optimization problems. In Dorigo et al. [2004], pages 294–301.

Randall, M. (2004). Near parameter free ant colony optimisation. In Dorigo et al. [2004], pages 374–381.

Reimann, M., Doerner, K., and Hartl, R. F. (2002). Insertion based ants for vehicle routing problems with backhauls and time windows. In Dorigo et al. [2002a], pages 135–148.

Reinelt, G. (1995). TSPLIB. http://elib.zib.de/pub/Packages/mp-testdata/tsp/tsplib/tsplib.html.

Rendl, F. and Wolkowicz, H. (1992). Applications of parametric programming and eigenvalue maximization to the quadratic assignment problem. *Mathematical Programming*, 53:63–78.

Resende, M. and Ribeiro, C. (2003). Greedy randomized adaptive search procedures. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers.

Ritchie, G. (2003). Static multi-processor scheduling with ant colony optimisation and local search. Master's thesis, University of Edinburgh.

Ritchie, G. and Levine, J. (2003). A fast, effective local search for scheduling independent jobs in heterogeneous computing environments. In *PLANSIG 2003: Proceedings of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group*, Glasgow.

Ritchie, G. and Levine, J. (2004). A hybrid ant algorithm for solving heterogeneous multi-processor scheduling problems. In *PLANSIG 2004: Proceedings of the 23rd Workshop of the UK Planning and Scheduling Special Interest Group*, Cork.

Roli, A., Blum, C., , and Dorigo, M. (2001). Aco for maximal constraint satisfaction problems. In *Proceedings of the Fourth Metaheuristics International Conference*, pages 187–191, Porto, Portugal.

Rubenstein, R. and Kroese, D. P. (2004). *The Cross-Entropy Method*. Springer.

Sahni, S. and Gonzalez, T. (1976). P-complete approximation problems. *J. ACM*, 23(3):555–565.

Scheuermann, B., Guntsch, M., Middendorf, M., and Schmeck, H. (2004). Time-scattered heuristic for the hardware implementation of population-based aco. In Dorigo et al. [2004], pages 250–261.

Schoonderwoerd, R., Holland, O., Bruten, J., and Rothkrantz, L. (1996). Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5:169–207.

Shmygelska, A., Hernández, R. A., and Hoos, H. H. (2002). An ant colony optimization algorithm for the 2d hp protein folding problem. In Dorigo et al. [2002a], pages 40–53.

Silva, C. A., Runkler, T. A., da Costa Sousa, J. M., and Palm, R. (2002). Ant colonies as logistic processes optimizers. In Dorigo et al. [2002a], pages 76–87.

Smith, B. (2003). Constraint programming in practice: Scheduling a rehearsal. Research Report APES-67-2003.

Socha, K. (2003). The Influence of Run-Time Limits on Choosing Ant System Parameters. In Cantu-Paz et al., E., editor, *Proceedings of GECCO 2003 – Genetic and Evolutionary Computation Conference*, volume 2723 of *LNCS*, pages 49–60. Springer-Verlag.

Socha, K. (2004). Aco for continuous and mixed-variable optimization. In Dorigo et al. [2004], pages 25–36.

Socha, K., Knowles, J. D., and Sampels, M. (2002). A max-min ant system for the university course timetabling problem. In Dorigo et al. [2002a], pages 1–13.

Socha, K., Sampels, M., and Manfrin, M. (2003). Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art. In *Proceedings of EvoCOP 2003 – 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization*, volume 2611 of *LNCS*, pages 334–345. Springer-Verlag.

Steinberg, L. (1961). The backboard wiring problem: A placement algorithm. *SIAM Review*, 3(1):37–50.

Stützle, T. (1997). MAX-MIN Ant System for Quadratic Assignment Problems. Technical Report AIDA–97–4, FG Intellektik, TU Darmstadt, Germany.

Stützle, T. (1998a). An ant approach to the Flow Shop Problem. In *Proceedings of EUFIT'98*, pages 1560–1564, Aachen.

Stützle, T. (1998b). Parallelization strategies for Ant Colony Optimization. *Lecture Notes in Computer Science*, 1498:722–731.

Stützle, T. (2004). Aco-tsp. http://iridia.ulb.ac.be/ mdorigo/ACO/aco-code/public-software.html.

Stützle, T., den Besten, M., , and Dorigo, M. (2000). Ant colony optimization for the total weighted tardiness problem. In *Proceedings of PPSN-VI, Sixth International Conference on Parallel Problem Solving from Nature*, volume 1917 of *LNCS*, pages 611–620.

Stützle, T. and Dorigo, M. (1999a). ACO Algorithms for the Quadratic Assignment Problem. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization*. McGraw-Hill.

Stützle, T. and Dorigo, M. (1999b). ACO Algorithms for the Traveling Salesman Problem. In Miettinen, K., Makela, M., Neittaanmaki, P., and Periaux, J., editors,

Evolutionary Algorithms in Engineering and Computer Science, pages 163–183. Wiley.

Stützle, T. and Dorigo, M. (2001). An experimental study of the simple ant colony optimization algorithm. In *Proceedings of the, 2001 WSES International Conference on Evolutionary Computalution Computation (EC'01)*. WSES-Press International.

Stützle, T. and Dorigo, M. (2002). A short convergence proof for a class of aco algorithms. In *IEEE Transactions on Evolutionary Computation*, pages 6(4):358–365.

Stützle, T. and Hoos, H. (1997). The MAX-MIN Ant System and Local Search for the Traveling Salesman Problem. In *IEEE International Conference on Evolutionary Computation*.

Stützle, T. and Hoos, H. (1998a). Improvements on the Ant System: Introducing the MAX-MIN Ant System. In Smith, G., Steele, N., and Albrecht, R., editors, *Proceedings of Artificial Neural Nets and Genetic Algorithms 1997*, pages 245–249. Springer Verlag Wien.

Stützle, T. and Hoos, H. (1998b). *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, chapter The MAX-MIN Ant System and Local Search for Combinatorial Optimization Problems: Towards Adaptive Tools for Combinatorial Global Optimization, pages 313–329. Kluwer Academic Publishers.

Stützle, T. and Hoos, H. H. (2000). MAX-MIN Ant System. In *Future Generation Computer Systems*, volume 16, pages 889–914.

Taillard, E. D. (1995). Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 3:87–455.

Taillard, É. D. and Gambardella, L. (1997a). Adaptive memories for the quadratic assignment problems. Technical Report IDSIA-87-97, IDSIA.

Taillard, É. D. and Gambardella, L. (1997b). An ant approach for structured Quadratic Assignment Problems. Technical Report IDSIA-22-97, IDSIA.

van der Zwaan, S. and Marques, C. (1999). Ant Colony Optimisation for job shop scheduling. S. van der Zwaan, and C. Marques. Ant Colony Optimisation for Job Shop Scheduling. Proceedings of the Third Workshop on Genetic Algorithms and Artificial Life (GAAL 99), 1999.

van Hemert, J. and Solnon, C. (2004). A study into ant colony optimization, evolution-

ary computation and constraint programming on binary constraint satisfaction problems. In *Evolutionary Computation in Combinatorial Optimization*, pages 114–123. Springer-Verlag.

van Laarhoven, P. J. M. (1987). *Simulated annealing : theory and applications*, chapter 2, page 10. Kluwer Academic Publishers, Dordrecht.

Varela, G. N. and Sinclair, M. C. (1999). Ant Colony Optimisation for virtual-wavelength-path routing and wavelength allocation. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 1809–1816, Mayflower Hotel, Washington D.C., USA. IEEE Press.

Vecerina, I. (2004). Re: Quality of rand(). comp.lang.c++.

Vittori, K., Gautrais, J., Araújo, A. F. R., Fourcassié, V., and Theraulaz, G. (2004). Modeling ant behavior under a variable environment. In Dorigo et al. [2004], pages 190–201.

Wedde, H., Farooq, M., and Zhang, Y. (2004). Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In Dorigo et al. [2004], pages 83–94.

White, T., Pagurek, B., and Oppacher, F. (1998). Connection management using adaptive agents. In Arabnia, H. R., editor, *Proc. of the 1998 Int'l Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA '98)*, pages 802–809, Las Vegas. CSREA Press.

Williamson, D. P. (1989). Analysis of the held-karp heuristic for the traveling salesman problem. Master's thesis, Massachusetts Institute of Technology.

Zhang, Y., Kuhn, L. D., and Fromherz, M. P. J. (2004). Improvements on ant routing for sensor networks. In Dorigo et al. [2004], pages 154–165.

Zlochin, M. and Dorigo, M. (2002). Model-based search for combinatorial optimization: A comparative study. In Guervós, J. J. M., Adamidis, P., Beyer, H.-G., Martín, J. L. F.-V., and Schwefel, H.-P., editors, *PPSN*, volume 2439 of *Lecture Notes in Computer Science*, pages 651–664. Springer.